DEPARTMENT OF DEFENCE

DEFENCE SCIENCE & TECHNOLOGY ORGANISATION

DSTO

A Replacement Database for the CH - 47D Spectrometric Oil Analysis Program

Paul Marsden and Andrew Becker

DSTO-TN-0412

A Replacement Database for the CH-47D Spectrometric Oil Analysis Program

Paul Marsden and Andrew Becker

Airframes and Engines Division
Aeronautical and Maritime Research Laboratory

DSTO-TN-0412

ABSTRACT

A Spectrometric Oil Analysis Program (SOAP) operates on selected Australian Defence Force platforms to assist in the prediction of incipient machinery failure. Historically, the data from the Australian Army CH-47D helicopters has been stored on a simple Microsoft Excel spreadsheet. DSTO was tasked by the Army Aircraft Logistic Management Squadron to assess the usefulness of this database. This report contains a detailed description of the replacement SOAP database designed by DSTO.

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION DSTO

Published by

DSTO Aeronautical and Maritime Research Laboratory
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999
© Commonwealth of Australia 2002
AR-012-161
March 2002

A Replacement Database for the CH-47D Spectrometric Oil Analysis Program

Executive Summary

This report contains a detailed description of a replacement Spectrometric Oil Analysis Program (SOAP) database that has been designed and developed by DSTO for use with the Australian Army CH-47D helicopter fleet.

Spectrometric Oil Analysis (SOA) is conducted on gearbox, engine and hydraulic oil samples that are taken periodically from all aircraft in the CH-47D fleet. The trends gained from the SOA can assist with the detection of incipient failure of mechanical components (bearings, gears etc). As mechanical components wear they shed small particles of metal that become entrained in the oil. As a wear related failure progresses, the quantity of particulate entrained in the oil increases. SOA is used to detect increasing trends for particular elements and hence assist in identifying the component prior to catastrophic failure.

CH-47D SOA samples are analysed by a local contractor in Townsville. The Army Aircraft Logistic Management Squadron (AALM SQN) and 'C' Squadron maintain hardcopies of the reports produced by the contractor. Historically, the AALM SQN also used a simple Microsoft Excel spreadsheet to collate and record all SOAP data in an attempt to identify trends. The format of this spreadsheet was not user-friendly and was found to contain significant structural flaws.

After assessing the condition of the old database, it was decided that a completely new database was required. Microsoft Excel was retained as the underlying program, however, a Visual Basic interface was created that made entering and viewing data easier for untrained operators. Another advantage was that no new software licenses needed to be purchased or maintained since Excel was an existing part of the computer network at AALM SQN. The DSTO-designed database has been commissioned at the AALM SQN.

Contents

1. Introduction

DSTO was tasked by the Army Aircraft Logistic Management Squadron (AALM SQN) to assess the usefulness of an existing Spectrometric Oil Analysis Program (SOAP) database. The database is used to store and trend SOAP data pertaining to CH-47D helicopters. Upon initial examination, the existing database was found to contain some significant structural flaws and did not have a user-friendly interface. A new database written in Microsoft Excel, and encompassing a Visual Basic user-interface, was produced for the AALM SQN. This report describes the functionality of the DSTO-designed SOAP database as well as the Visual Basic user-interface program.

2. Background

2.1 Spectrometric Oil Analysis (SOA)

Spectrometric Oil Analysis (SOA) is an analytical technique for identifying the elemental composition of particles (up to approximately 8 micron) entrained in machinery oil samples. As mechanical components wear they shed small metallic particles that become entrained in the oil. As a wear related failure initiates and then progresses, the quantity of particular elements increases and this can be observed in the SOA trends. Knowledge of the constituent metals in a particular system is then used to determine the likely origin of the wear particles. Original Equipment Manufacturers (OEMs) of aircraft usually set quantity and rate-of-increase limits for each element.

SOA is conducted on gearbox oil (5 samples/aircraft), engine oil (2 samples/aircraft) and hydraulic oil (3 samples/aircraft) samples that are taken every 25 airframe hours from all aircraft in the Australian Army CH-47D fleet. The 100 ml samples are sent to a local contractor where spectrometric oil analysis is conducted. Hardcopies of the contractor reports are kept by 'C' Squadron and the AALM SQN. In addition to hardcopies, AALM SQN personnel maintain an electronic SOAP database. This database was intended to enable fleet wide trends to be observed and comparison within the fleet to be made.

2.2 The Original SOAP Database

The original SOAP database was an Excel spreadsheet with a limited plotting capability. Unfortunately the database had been set up to trend results based on airframe tail number. For a SOAP to be meaningful the major assemblies (gearboxes, engines etc) should be individually trended. This is possible since all major assemblies have an individual serial number. Trending by serial number enables incipient faults within a particular assembly to be correctly identified. Trending based on tail number can lead to inaccurate diagnosis or confusion since the major assemblies are periodically replaced.

2.3 DSTO-Designed SOAP Database

After assessing the condition of the existing SOAP database, it was decided to create a new and improved database. It was also decided to use the existing AALM SQN network version of Microsoft Excel as the database foundation since this would avoid the need to purchase new software. It was also felt that personnel would be more likely to feel comfortable using the database if it was a familiar program.

An important element of the new database was the inclusion of a Visual Basic user-interface. A detailed description of the Visual Basic program is contained in Appendix A. The interface greatly simplifies the input of new data, the viewing of trends and the manipulation of data as assemblies are fitted to or removed from various aircraft.

The trend plotting function was significantly enhanced in the new database by the amalgamation of oil top-up quantities and SOA data on a single trend plot. This enables easy correlation between trend fluctuations and oil top-up quantities.

2.3.1 Introduction Screen

Figure 1 shows the first screen that appears when the database is opened. The five large buttons on the right enable the user to:

1. Enter SOAP data for the aircraft tail number currently selected.
2. View SOA trends for aircraft transmissions and hydraulic systems.
3. Add a new transmission serial number to the database.
4. Remove and replace transmission components in aircraft.
5. Set MARGINAL and ABNORMAL alarm limits for each transmission type.

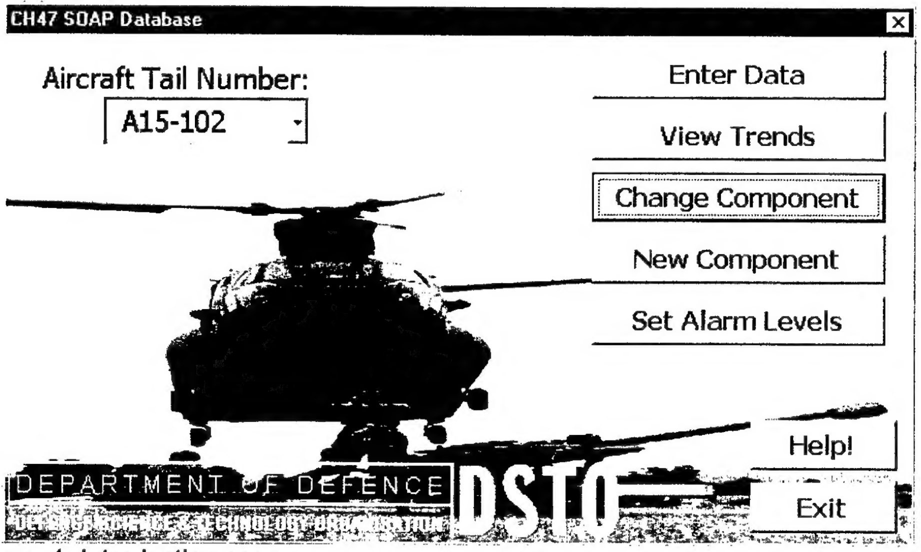
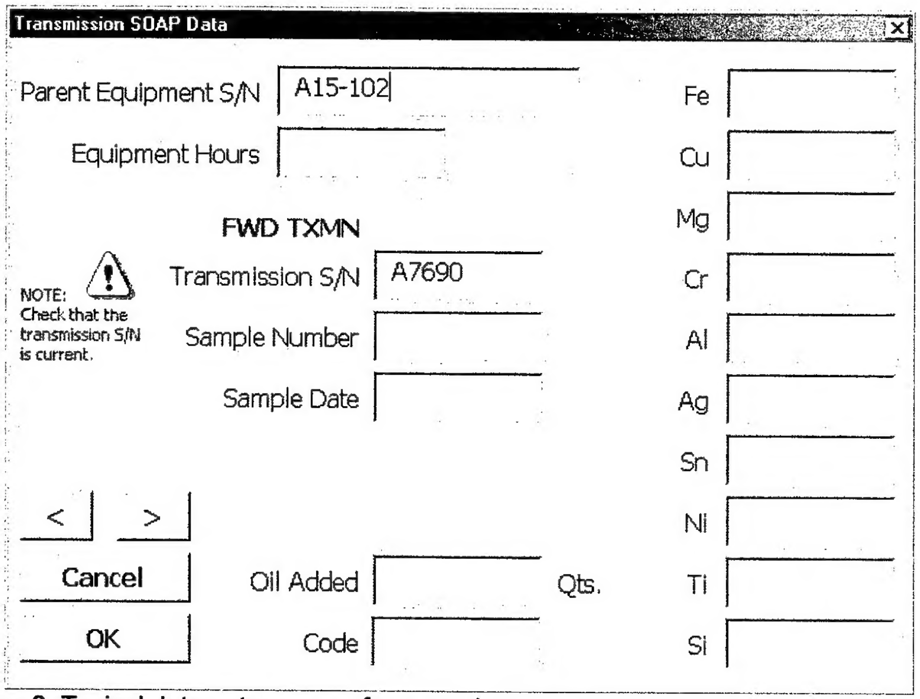


Figure 1. Introduction screen

2.3.2 Data Input Screens

There are separate data entry screens for each of the five transmissions (gearboxes), both engines and each of the three hydraulic systems per aircraft. Figure 2 shows a typical data entry screen for a gearbox. The hydraulic systems have measurements for water and viscosity in addition to the SOA elements that are listed down the right side of the screen.

In both cases the SOAP data will be trended by hours: component hours for transmissions, and aircraft hours for the hydraulic systems. The database will therefore not accept an entry unless the *Equipment Hours* field has been filled in.



Parent Equipment S/N	A15-102
Equipment Hours	
Transmission S/N	A7690
Sample Number	
Sample Date	
Oil Added	
Code	

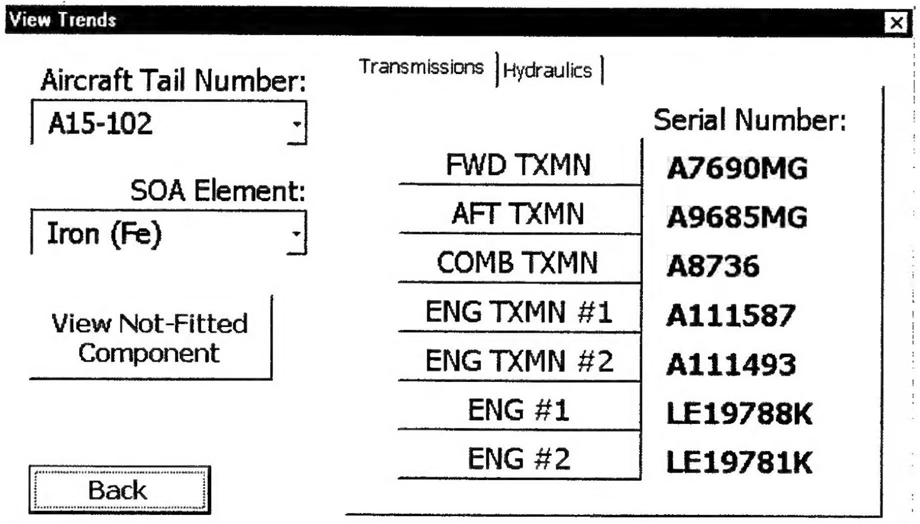
Figure 2. Typical data entry screen for a gearbox

It is ESSENTIAL that the correct transmission serial number is selected for each aircraft. Installed component serial numbers can be altered via the *Change Component* form (Section 2.3.4).

2.3.3 View Trends Screen

From the *View Trends* screen (Figure 3) the SOA trends for any installed transmission component or hydraulic system may be selected. The trends may be shown on a single plot showing the results for one element in a particular component. By selecting *All Aircraft* from the *Aircraft Tail Number* list, the results for one SOA element in all the currently installed transmission components of one type may be viewed simultaneously.

SOA trends for transmission components that are not currently fitted to any aircraft may be viewed by selecting the *View Not-Fitted Component* button. Once this button has been pushed, the specific component can be selected from a list of all components not currently fitted to aircraft.



Aircraft Tail Number: A15-102

SOA Element: Iron (Fe)

	Serial Number:
FWD TXMN	A7690MG
AFT TXMN	A9685MG
COMB TXMN	A8736
ENG TXMN #1	A111587
ENG TXMN #2	A111493
ENG #1	LE19788K
ENG #2	LE19781K

Figure 3. View Trends screen

Figure 4 shows an example of a SOA trend plot for a single component; in this case it shows the trend for iron in the forward gearbox. The thin vertical bars indicate the amount of top-up oil that has been added and are measured against the right hand vertical scale. The two thick dashed lines indicate the quantity alarm levels for the chosen element in this transmission. The lower line is the MARGINAL level and the upper line is the ABNORMAL level. The two buttons on the right hand side of the plot take the user back to the *View Trends* screen or to a print preview screen.

If the *All Aircraft* option has been selected a separate trend plot will be produced for the selected component from each aircraft. Each of these trend plots will have the same form as that shown below and will be displayed on a single page.

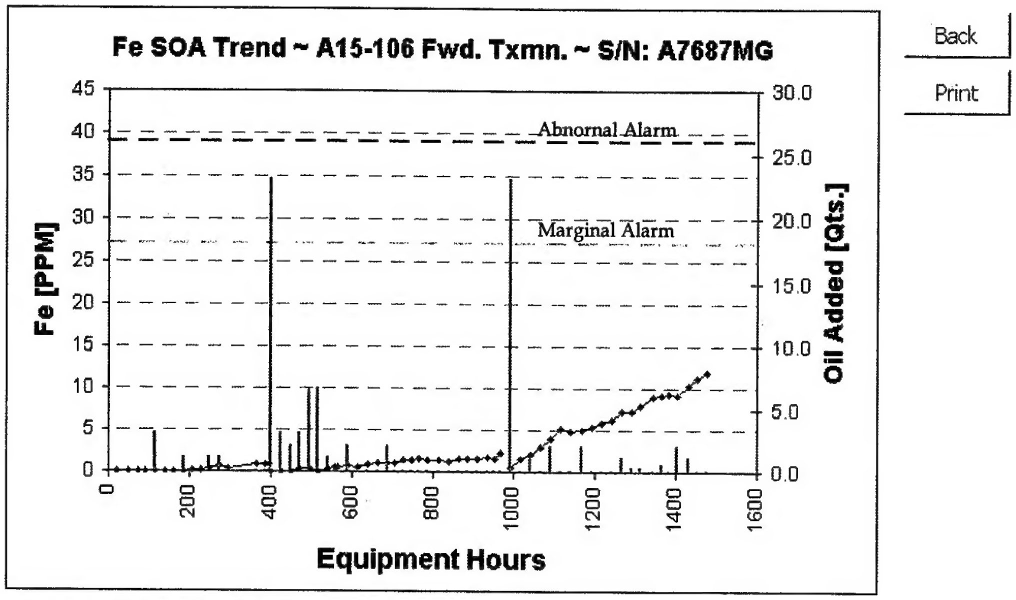
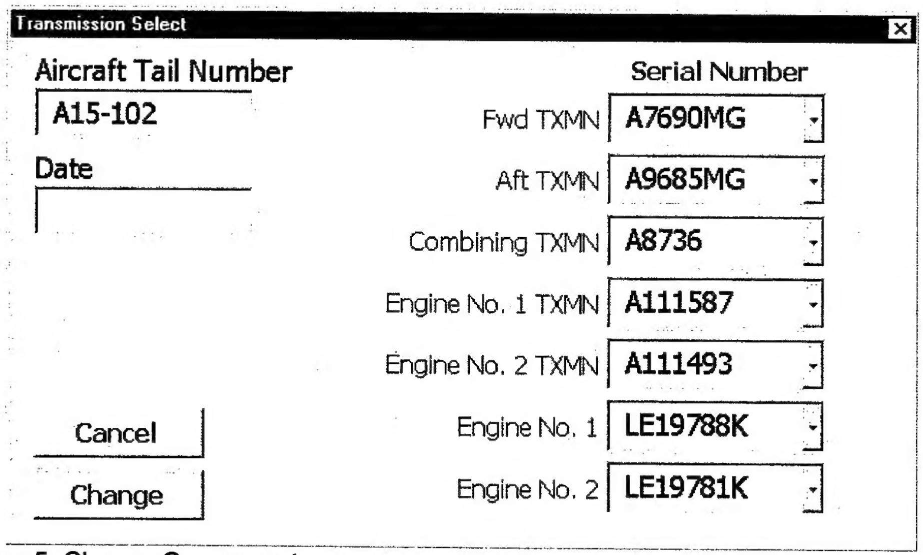


Figure 4. Example of a SOA trend plot

2.3.4 Change Component Screen

Any SOAP data that is entered for a particular aircraft tail number is stored with the component serial numbers that are currently assigned to that aircraft. In the event that a transmission is removed from an aircraft or is replaced, the database must be updated to reflect this change using the *Component Change* screen (Figure 5).


Transmission Select

Aircraft Tail Number: A15-102

Date:

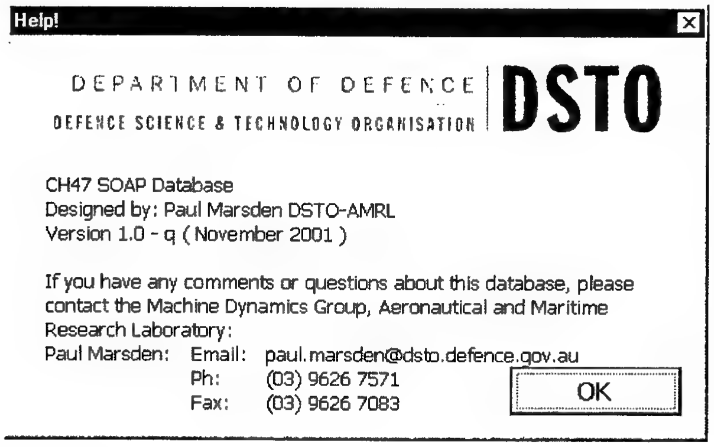
	Serial Number
Fwd TXMN	A7690MG
Aft TXMN	A9685MG
Combining TXMN	A8736
Engine No. 1 TXMN	A111587
Engine No. 2 TXMN	A111493
Engine No. 1	LE19788K
Engine No. 2	LE19781K

Figure 5. Change Component screen

The *Change Component* screen displays a list of available components for each of the seven locations on the aircraft. The component number initially shown on the list is the one currently attributed to that aircraft by the database. The dropdown menus then list replacement options from those components that are currently not fitted to any aircraft. If the component has not yet been replaced there is also a *Removed* option. This indicates that the aircraft does not currently have a component fitted in that location. In either case, the component that was initially fitted in that position then becomes *Not-Fitted*. Trends for such components may be accessed via the *View Not-Fitted Component* option on the *View Trends* screen (Section 2.3.3).

2.3.5 Help Screen

The Help screen (Figure 6) was included so that problems or questions about the database could be directed to DSTO for resolution.



CH47 SOAP Database
Designed by: Paul Marsden DSTO-AMRL
Version 1.0 - q (November 2001)

If you have any comments or questions about this database, please contact the Machine Dynamics Group, Aeronautical and Maritime Research Laboratory:

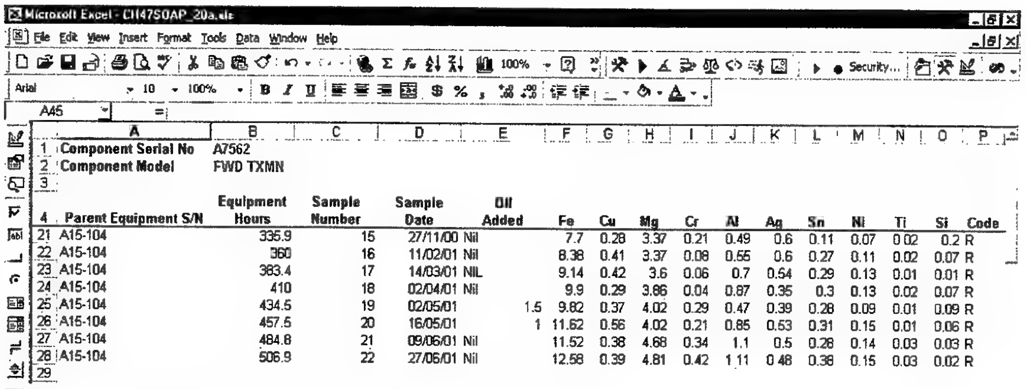
Paul Marsden: Email: paul.marsden@dsto.defence.gov.au
Ph: (03) 9626 7571
Fax: (03) 9626 7083

Figure 6. Help screen

2.4 Database Structure

Each component is allocated a separate worksheet in the Excel file (named by serial number) where the data for that particular component is stored. An example of a data sheet is shown in figure 7. During the normal operation of the database these sheets will remain hidden from the operator.

A master list of all component serial numbers and their type is kept on a sheet named *Transmissions*. A separate sheet called *Aircraft* contains the serial numbers of the transmissions that are currently fitted to each aircraft. A transmission that appears in the master list, but is not currently associated with any particular aircraft is considered to be *Not-Fitted*. The *Transmissions* and *Aircraft* sheets are also hidden from the operator during normal operation of the database.



Microsoft Excel - CH47SOAP_20a.xls

A45 =

1	Component Serial No	A7562														
2	Component Model	FWD TXMN														
3																
4	Parent Equipment S/N	Equipment Hours	Sample Number	Sample Date	Oil Added	Fe	Cu	Mg	Cr	Al	Ag	Sn	Ni	Ti	Si	Code
21	A15-104	335.9	15	27/11/00	Nil	7.7	0.28	3.37	0.21	0.49	0.6	0.11	0.07	0.02	0.2	R
22	A15-104	360	16	11/02/01	Nil	8.38	0.41	3.37	0.08	0.55	0.6	0.27	0.11	0.02	0.07	R
23	A15-104	383.4	17	14/03/01	NIL	9.14	0.42	3.6	0.06	0.7	0.54	0.29	0.13	0.01	0.01	R
24	A15-104	410	18	02/04/01	Nil	9.9	0.29	3.86	0.04	0.87	0.35	0.3	0.13	0.02	0.07	R
25	A15-104	434.5	19	02/05/01	1.5	9.82	0.37	4.02	0.29	0.47	0.39	0.28	0.09	0.01	0.09	R
26	A15-104	457.5	20	16/05/01	1	11.62	0.56	4.02	0.21	0.85	0.53	0.31	0.15	0.01	0.06	R
27	A15-104	484.8	21	09/06/01	Nil	11.52	0.38	4.68	0.34	1.1	0.5	0.28	0.14	0.03	0.03	R
28	A15-104	506.9	22	27/06/01	Nil	12.58	0.39	4.81	0.42	1.11	0.48	0.38	0.15	0.03	0.02	R
29																

Figure 7. Example of a data sheet for a component

This structure enables the SOAP trends for a component to be followed, even in the event that it is removed from one aircraft and fitted to another. Since each installed component is associated with a particular aircraft, all of the components on an aircraft are accessible for both data entry and trend viewing.

2.5 Other Applications

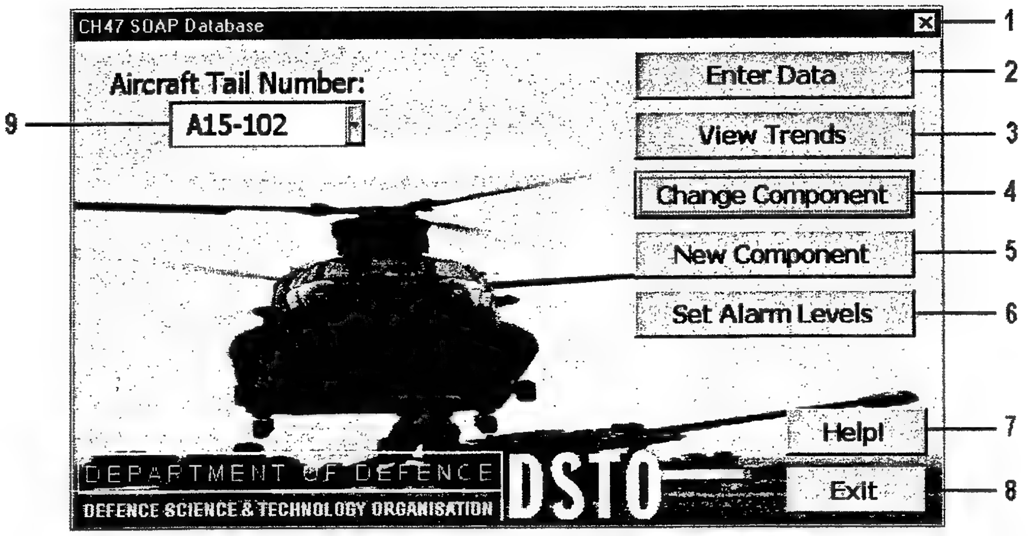
The DSTO-designed SOAP database can be readily applied to other aircraft types as required. The basic structure of the database can also be applied to other condition monitoring functions. An example of this is the application of the core database to oil condition monitoring for the Royal Australian Air Force TF30 engines. The database has been modified so that oil chemical parameters can be easily input and trends viewed. It is intended that this version reside on a web page server and hence will allow the laboratory chemist to input the data whilst simultaneously allowing squadron and support personnel (located interstate) to view the data.

3. Conclusion

This report has described a DSTO-designed SOAP database intended for use by (but not limited to) the Australian Army CH-47D helicopter fleet. This database is being used by the AALM SQN and has replaced an original database that contained some structural flaws. The improved functionality of the new database has been described and the user-interface program has been documented in detail (Appendix A) for future reference.

Appendix A: Visual Basic User-Interface Program Description

A.1. Main form: fmAircraft



```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub change_component_Click()

    Application.ScreenUpdating = False
    fmAircraft.Hide
    Load fmTransmission
    fmTransmission.Initialize
    fmTransmission.Show
End Sub
```

NOTES:
Called when ***Change Component*** button (5) is pressed. Displays the form fmTransmission - used to change transmissions into and out of aircraft.

```
Private Sub Levels_Click()
    fmAircraft.Hide
    Load fmLevels
    fmLevels.Initialize
    fmLevels.Show
End Sub
```

NOTES:
Called when *Set Alarm Levels* button (6) is pressed. Displays the form fmLevels which is used to set the MARGINAL and ABNORMAL alarm levels.

```
Private Sub enter_data_Click()
    Dim AC_row As Integer

    fmAircraft.Hide

    AC_row = tail_no.ListIndex + 2   ' AC data starts at row 2
    fmChooseTX.tail_no.ControlSource = "Aircraft!A" & AC_row

    Load fmChooseTX
    fmChooseTX.Show
End Sub
```

NOTES:
Called when the *Enter Data* button (2) is pressed. Displays the form fmChooseTX and loads the selected aircraft tail number from the list-box (9).

```
Private Sub EXIT_Button_Click()
    Unload fmAircraft
    ActiveWorkbook.Close
End Sub
```

NOTES:
Called when the *Exit* button (8) is pressed. Closes the workbook – if changes have been made the option will be given to save before exit.

```
Sub Initialize()
    Application.ScreenUpdating = False
    Sheets("Aircraft").Visible = True

    ' Populate the aircraft tail number list box
    ' from the list of all aircraft tail numbers.
    Sheets("Aircraft").Activate
    ActiveSheet.Range("A2").Select

    While (ActiveCell.Value <> "")
        tail_no.AddItem ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
    Wend
    tail_no.ListIndex = 1
    Sheets("Aircraft").Visible = False
    Application.ScreenUpdating = True
End Sub
```

NOTES:
Initialises data displayed on the form fmAircraft.

```
Private Sub help_button_Click()
    Load fmHelp
    fmHelp.Show
End Sub
```

NOTES:
Called when the *Help* button (7) is pressed. Displays the form fmHelp.

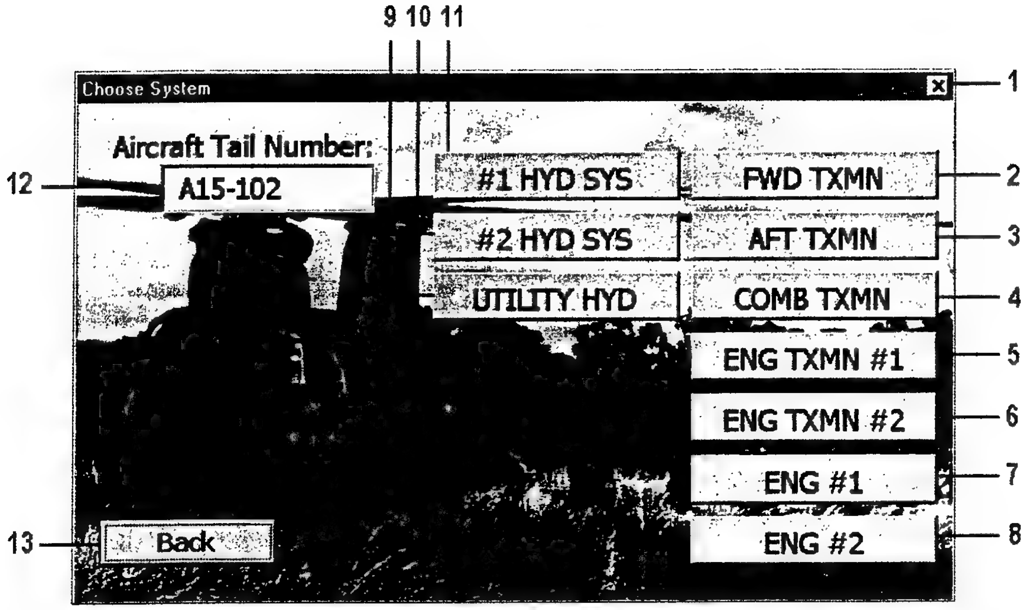
```
Private Sub view_trends_Click()
    fmAircraft.Hide
    Load fmTrends
    fmTrends.Initialize
    fmTrends.Show
End Sub
```

NOTES:
Called when the *View Trends* button (3) is pressed. Displays the form fmTrends.

```
Private Sub new_component_Click()
    fmAircraft.Hide

    Load fmNewComponent
    fmNewComponent.Initialize
    fmNewComponent.Show
End Sub
```

NOTES:
Called when the *New Component* button (5) is pressed. Displays the form fmTrends.

A.2. System choice form (Enter Data): fmChooseTX



```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub aft_txmn_Click()
    fmChooseTX.Hide
    Call SOAP_data("AFT TXMN", "C")
End Sub
```

NOTES:
Called when *AFT TXMN* button (3) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected aft transmission.

```
Private Sub comb_txmn_Click()
    fmChooseTX.Hide
    Call SOAP_data("COMB TXMN", "D")
End Sub
```

NOTES:
Called when *COMB TXMN* button (4) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected combining transmission.

```
Private Sub eng_txmn1_Click()
    fmChooseTX.Hide
    Call SOAP_data("ENG TXMN 1", "E")
End Sub
```

NOTES:
Called when *ENG TXMN #1* button (5) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected engine transmission.

```
Private Sub eng_txmn2_Click()
    fmChooseTX.Hide
    Call SOAP_data("ENG TXMN 2", "F")
End Sub
```

NOTES:
Called when *ENG TXMN #2* button (6) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected engine transmission.

```
Private Sub eng1_Click()
    fmChooseTX.Hide
    Call SOAP_data("ENG 1", "G")
End Sub
```

NOTES:
Called when *ENG #1* button (7) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected engine.

```
Private Sub eng2_Click()
    fmChooseTX.Hide
    Call SOAP_data("ENG 2", "H")
End Sub
```

NOTES:
Called when *ENG #2* button (14) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected engine.

```
Private Sub fwd_txmn_Click()
    fmChooseTX.Hide
    Call SOAP_data("FWD TXMN", "B")
End Sub
```

NOTES:
Called when *FWD TXMN* button (2) is pressed. Calls the SOAP_data function to initialise and display the data entry form for the selected forward transmission.

```
Private Sub HYD_data(caption As String, HYD_col As String)
    Dim HYD_row As Integer

    Application.ScreenUpdating = False
    Load fmHydraulics
    fmHydraulics.tail_no.Value = fmAircraft.tail_no.Value
    fmHydraulics.HYD_lable.caption = caption

    Sheets(fmAircraft.tail_no.Value).Visible = True
    Sheets(fmAircraft.tail_no.Value).Activate

    HYD_row = 5 ' Hydraulics data starts on row 5
    ActiveSheet.Range(HYD_col & HYD_row).Select
    While (ActiveCell.Value <> "")
        HYD_row = HYD_row + 1
        ActiveCell.Offset(1, 0).Select
    Wend
    fmHydraulics.SetControls

    Call fmHydraulics.Initialize(HYD_row, HYD_col)

    Sheets(fmAircraft.tail_no.Value).Visible = False
    Application.ScreenUpdating = True
    fmHydraulics.Show
End Sub
```

NOTES:
Initialises the data entry page for hydraulic systems. Is passed a string describing the system (for display) and the column in which the data for that system is stored.

```
Private Sub no1_hyd_sys_Click()
    fmChooseTX.Hide
    Call HYD_data("No. 1 HYD. SYS.", "A")
End Sub
```

NOTES:
Called when *#1 HYD SYS* button (11) is pressed. Calls the HYD_data function to initialise and display the data entry form for the selected hydraulic system.

```
Private Sub no2_hyd_sys_Click()
    fmChooseTX.Hide
    Call HYD_data("No. 2 HYD. SYS.", "R")
End Sub
```

NOTES:
Called when *#2 HYD SYS* button (10) is pressed. Calls the HYD_data function to initialise and display the data entry form for the selected hydraulic system.

```
Private Sub utility_hyd_Click()
    fmChooseTX.Hide
    Call HYD_data("UTILITY HYD.", "AI")
End Sub
```

NOTES:
Called when *UTILITY HYD* button (9) is pressed. Calls the HYD_data function to initialise and display the data entry form for the selected hydraulic system.

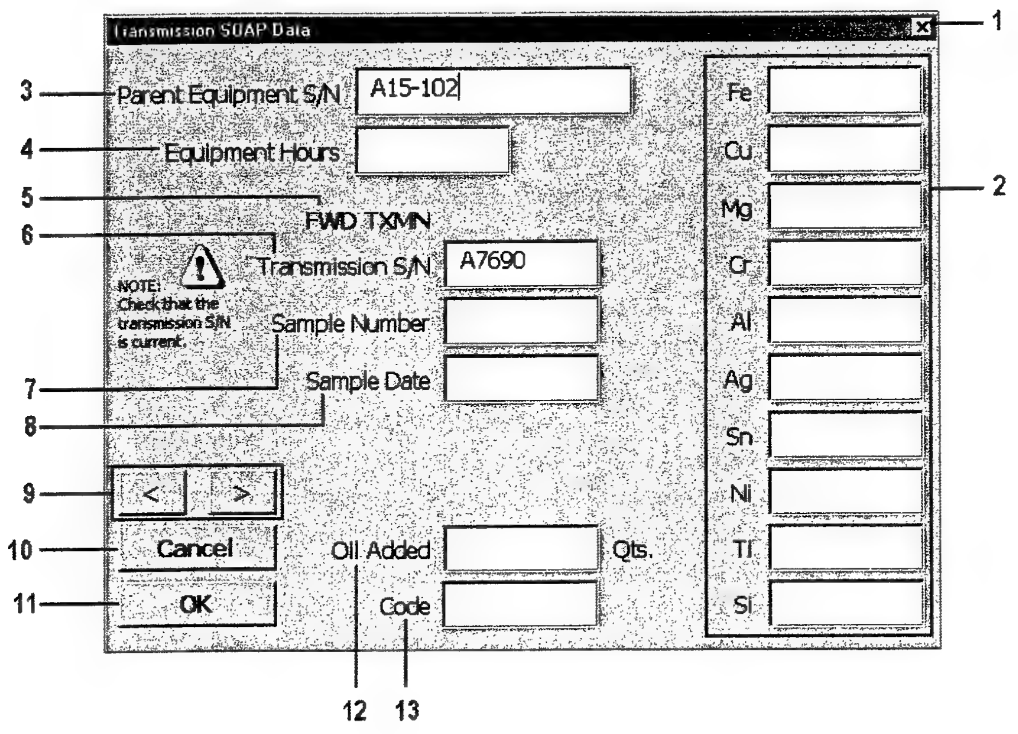
```
Private Sub SOAP_data(caption As String, TX_col As String)
    Dim AC_row As Integer
    Dim SOAP_row As Integer
    Dim TXMN As String
    Dim response As String

    Application.ScreenUpdating = False

    ' AC data starts at row 2
    AC_row = fmAircraft.tail_no.ListIndex + 2
    Load fmSoapData

    Sheets("Aircraft").Visible = True

    ' Get transmission type and serial number
    fmSoapData.TX_lable.caption = caption
    fmSoapData.transmission.ControlSource = …
      …"Aircraft!" & TX_col & AC_row
    Sheets("Aircraft").Activate
    Range(TX_col & AC_row).Select
    TXMN = ActiveCell.Value

    If (TXMN <> "REMOVED") Then
       SOAP_row = 5     ' SOAP Data starts on row 5
       Sheets(TXMN).Visible = True
       Sheets(TXMN).Activate
       ActiveSheet.Range("A" & SOAP_row).Select

       ' Find last empty row on data-sheet.
       While (ActiveCell.Value <> "")
           SOAP_row = SOAP_row + 1
           ActiveCell.Offset(1, 0).Select
       Wend

       fmSoapData.tail_no.ControlSource = "A" & SOAP_row
       Sheets("Aircraft").Activate
       ActiveSheet.Range("A" & AC_row).Select
       fmSoapData.tail_no.Value = ActiveCell.Value
       Sheets(TXMN).Visible = False

       Call fmSoapData.Initialize(SOAP_row, TXMN)
       Sheets("Aircraft").Visible = False
       fmSoapData.SetControls
       fmSoapData.Show
    Else
        response = MsgBox("This transmission has been removed.",…
                     … vbOKOnly + vbExclamation + vbApplicationModal)
        fmChooseTX.Show
    End If
    Sheets("Aircraft").Visible = False
    Application.ScreenUpdating = True

End Sub
```

NOTES:
Initialises the data entry page for transmissions. Is passed a string describing the transmission (for display) and the column in which the data for that system is stored.

```
Private Sub back_Click()
    Unload fmChooseTX
    fmAircraft.Show
End Sub
```

NOTES:
Called when *Back* button (13) is pressed. Returns to the main form, fmAircraft.

A.3. Enter transmission SOAP data form: fmSoapData



Parent Equipment S/N A15-102
Equipment Hours
FWD TXMN
Transmission S/N A7690
NOTE: Check that the transmission S/N is current.
Sample Number
Sample Date
Oil Added Qts.
Code

```
Dim SOAP_row As Integer
Dim MIN_row As Integer
Dim MAX_row As Integer
Dim the_sheet As String
```

NOTES:
Variables local to the form fmSoapData

```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Sub SetControls()
    Application.ScreenUpdating = False
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    Range("D" & SOAP_row).Select
    ActiveCell.NumberFormat = "@"

    fmSoapData.tail_no.ControlSource = "A" & SOAP_row

    fmSoapData.hours.ControlSource = "B" & SOAP_row
    fmSoapData.sample_no.ControlSource = "C" & SOAP_row
    fmSoapData.sample_date.ControlSource = "D" & SOAP_row
    fmSoapData.oil_added.ControlSource = "E" & SOAP_row

    fmSoapData.Fe.ControlSource = "F" & SOAP_row
    fmSoapData.Cu.ControlSource = "G" & SOAP_row
    fmSoapData.Mg.ControlSource = "H" & SOAP_row
    fmSoapData.Cr.ControlSource = "I" & SOAP_row
    fmSoapData.Al.ControlSource = "J" & SOAP_row
    fmSoapData.Ag.ControlSource = "K" & SOAP_row
    fmSoapData.Sn.ControlSource = "L" & SOAP_row
    fmSoapData.Ni.ControlSource = "M" & SOAP_row
    fmSoapData.Ti.ControlSource = "N" & SOAP_row
    fmSoapData.Si.ControlSource = "O" & SOAP_row

    fmSoapData.code.ControlSource = "P" & SOAP_row
    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
This routine links the appropriate data in the worksheet to the textboxes displayed on the form (2, 4, 7, 8, 12, 13). It is required when the scroll forward and scroll back features are used.

```
Private Sub forward_Click()
    Application.ScreenUpdating = False
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    If (SOAP_row < MAX_row) Then
        SOAP_row = SOAP_row + 1
    End If

    SetControls
    ActiveSheet.Range("D" & SOAP_row).Select
    reformat_dates

    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
Called when the ">" button (9) is pressed. Displays the data from the next entry for the current component in the database. The data displayed may be edited.

```
Sub reformat_dates()
    Dim the_date As Date
    Dim tmp_day As String
    Dim tmp_month As String
    Dim tmp_year As String
    Dim tmp_date As String

    If ActiveCell.Value <> Empty Then
        the_date = ActiveCell.Value
        tmp_day = Day(the_date)
        tmp_month = Month(the_date)
        tmp_year = Year(the_date)

        ' Reformat to give day-month-year
        Select Case tmp_month
            Case "1"
                tmp_month = "Jan"
            Case "2"
                tmp_month = "Feb"
            Case "3"
                tmp_month = "Mar"
            Case "4"
                tmp_month = "Apr"
            Case "5"
                tmp_month = "May"
            Case "6"
                tmp_month = "Jun"
            Case "7"
                tmp_month = "Jul"
            Case "8"
                tmp_month = "Aug"
            Case "9"
                tmp_month = "Sep"
            Case "10"
                tmp_month = "Oct"
            Case "11"
                tmp_month = "Nov"
            Case "12"
                tmp_month = "Dec"
        End Select

        tmp_date = tmp_day & "/" & tmp_month & "/" & tmp_year
        ActiveCell.NumberFormat = "@"
        ActiveCell.FormulaR1C1 = tmp_date
        ActiveCell.NumberFormat = "dd/mm/yy"
        ActiveCell.NumberFormat = "@"
    End If
End Sub
```

NOTES:
Used to reformat the date entered, forcing it to stay in the Australian format: (day/month/year).

```
Sub Initialize(the_row As Integer, the_soap_sheet As String)
    Application.ScreenUpdating = False

    the_sheet = the_soap_sheet
    SOAP_row = the_row
    MIN_row = 5 ' Data starts at row 5
    the_sheet = transmission.Value
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    MAX_row = MIN_row
    ActiveSheet.Range("A" & MIN_row).Select
    While (ActiveCell.Value <> "")
        MAX_row = MAX_row + 1
        ActiveCell.Offset(1, 0).Select
    Wend

    Range("A" & SOAP_row).Select
    ActiveCell.Value = fmAircraft.tail_no.Value
    SetControls

    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
Sets the form variables that describe which row is active and which component has been selected. The sheet corresponding to the selected transmission serial number is made current. Determines where the last data entry for the selected system is located – new data is entered on this row (SOAP_row).

```
Private Sub Cancel_Click()
    Dim i As Integer

    Application.ScreenUpdating = False
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    ActiveSheet.Range("A" & MAX_row & ":P" & MAX_row).Select
    Selection.Delete

    For i = 5 To MAX_row - 1
        ActiveSheet.Range("D" & i).Select
        reformat_dates
    Next i

    Sheets(the_sheet).Visible = False

    Application.ScreenUpdating = True
    Unload fmSoapData
    fmChooseTX.Show
End Sub
```

NOTES:
Called when the *Cancel* button (10) is pressed. Exits from the SOAP data entry page without making a new entry and returns to the system selection page (fmChooseTX). Any changes made to previous data are retained.

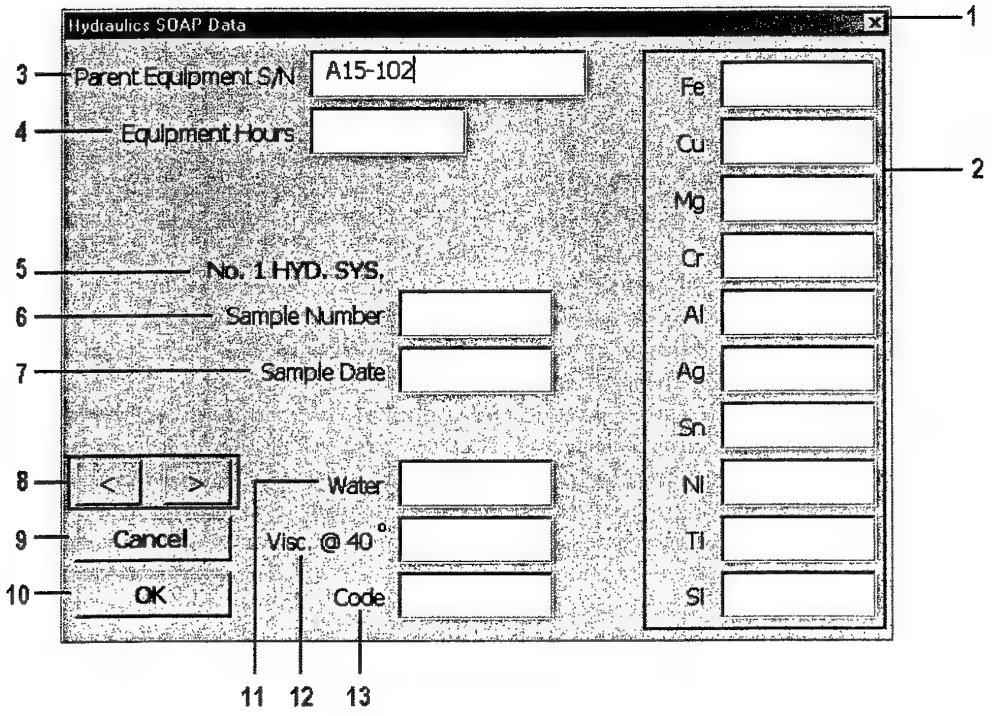
```
Private Sub OK_Button_Click()
    Dim hours_flag As Boolean
    Dim response As String

    Application.ScreenUpdating = False
    hours_flag = True

    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    ' Component hours from column B
    ActiveSheet.Range("B" & MAX_row).Select

    If ActiveCell.Value = Empty Then
        SOAP_row = MAX_row
        SetControls
        response = MsgBox("Please ensure component hours are …
                        … entered.", vbOKOnly + vbExclamation + …
                        … vbApplicationModal)
        hours_flag = False
    End If

    Sheets(the_sheet).Visible = False
    Application.ScreenUpdating = True

    If (hours_flag = True) Then
        Unload fmSoapData
        fmChooseTX.Show
    End If
End Sub
```

NOTES:
Called when the *OK* button (11) is pressed. Exits from the SOAP data entry page and returns to the system selection page fmChooseTX, A new entry is made in the database and any changes made to previous data are retained.

```
Private Sub back_Click()
    Application.ScreenUpdating = False

    If (SOAP_row > MIN_row) Then
        SOAP_row = SOAP_row - 1
    End If

    SetControls
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    ActiveSheet.Range("D" & SOAP_row).Select
    reformat_dates

    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
Called when the "<" button (9) is pressed. Displays the data from the previous entry for the current component in the database. The data displayed may be edited.

A.4. Enter hydraulics SOAP data form: fmHydraulics



```
Dim HYD_row As Integer
Dim HYD_col As String
Dim MIN_row As Integer
Dim MAX_row As Integer
Dim the_sheet As String
```

NOTES:
Variables local to the form fmHydraulics

```
Private Sub UserForm_QueryClose _
  (Cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        Cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub back_Click()
    Application.ScreenUpdating = False

    If (HYD_row > MIN_row) Then
        HYD_row = HYD_row - 1
    End If

    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    Range(HYD_col & HYD_row).Select
    SetControls

    Range(HYD_col & HYD_row).Select
    ActiveCell.Offset(0, 1).Select

    reformat_dates

    Sheets(the_sheet).Visible = False
    'Application.ScreenUpdating = True
End Sub
```

NOTES:
Called when the "<" button (8) is pressed. Displays the data from the previous entry for the current component in the database. The data displayed may be edited.

```
Private Sub Cancel_Click()
    Application.ScreenUpdating = False
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    ActiveSheet.Range(HYD_col & MAX_row).Select
    ActiveSheet.Range(HYD_col & MAX_row & ":" & …
      … ActiveCell.Offset(0, 16).Address).Select
    Selection.ClearContents

    Sheets(the_sheet).Visible = False
    Application.ScreenUpdating = True
    Unload fmHydraulics
    fmChooseTX.Show
End Sub
```

NOTES:
Called when the *Cancel* button (9) is pressed. Exits from the Hydraulics system data entry page without making a new entry and returns to the system selection page (fmChooseTX). Any changes made to previous data are retained.

```
Private Sub forward_Click()

    Application.ScreenUpdating = False

    If (HYD_row < MAX_row) Then
        HYD_row = HYD_row + 1
    End If

    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    Range(HYD_col & HYD_row).Select
    SetControls

    Range(HYD_col & HYD_row).Select
    ActiveCell.Offset(0, 1).Select

    reformat_dates

    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
Called when the ">" button (8) is pressed. Displays the data from the next entry for the current component in the database. The data displayed may be edited.

```
Sub Initialize(the_row As Integer, the_col As String)
    Application.ScreenUpdating = False

    HYD_row = the_row
    HYD_col = the_col
    MIN_row = 5 ' Data starts at row 5
    the_sheet = fmAircraft.tail_no.Value
    Sheets(the_sheet).Visible = True
    Sheets(the_sheet).Activate

    MAX_row = MIN_row
    ActiveSheet.Range(HYD_col & MIN_row).Select
    While (ActiveCell.Value <> "")
        MAX_row = MAX_row + 1
        ActiveCell.Offset(1, 0).Select
    Wend

    Sheets(the_sheet).Visible = False
End Sub
```

NOTES:
Sets the form variables that describe which row is active and which hydraulic system has been selected, these are used to refer to the correct data in the worksheets. Determines where the last data entry for the selected system is located.

```
Private Sub OK_Button_Click()
    Dim the_date As Date
    Dim tmp_sheet As String
    Dim tmp_day As String
    Dim tmp_month As String
    Dim tmp_year As String
    Dim tmp_date As String
    Dim tmp_row As Integer
    Dim tmp_col As String
    Dim i As Integer
    Dim hours_flag As Boolean
    Dim response As String

    hours_flag = True
    tmp_sheet = the_sheet
    tmp_row = MAX_row
    tmp_col = HYD_col

    Sheets(tmp_sheet).Visible = True
    Sheets(tmp_sheet).Activate

    ActiveSheet.Range(tmp_col & MAX_row).Select
    ActiveCell.Value = "X"

    ActiveSheet.Range(tmp_col & MAX_row).Select
    ActiveCell.Offset(0, 2).Select ' Select the hours column...
    If ActiveCell.Value = Empty Then
        HYD_row = MAX_row
        SetControls
        response = MsgBox("Please ensure component hours …
                    … are entered.", vbOKOnly + vbExclamation + …
                    … vbApplicationModal)
        hours_flag = False
    End If

    Sheets(tmp_sheet).Visible = False
    Application.ScreenUpdating = True

    If (hours_flag = True) Then
        Unload fmHydraulics
        fmChooseTX.Show
    End If
End Sub
```

NOTES:
Called when the *OK* button (10) is pressed. Exits from the Hydraulics system data entry page and returns to the system selection page fmChooseTX, A new entry is made in the database and any changes made to previous data are retained

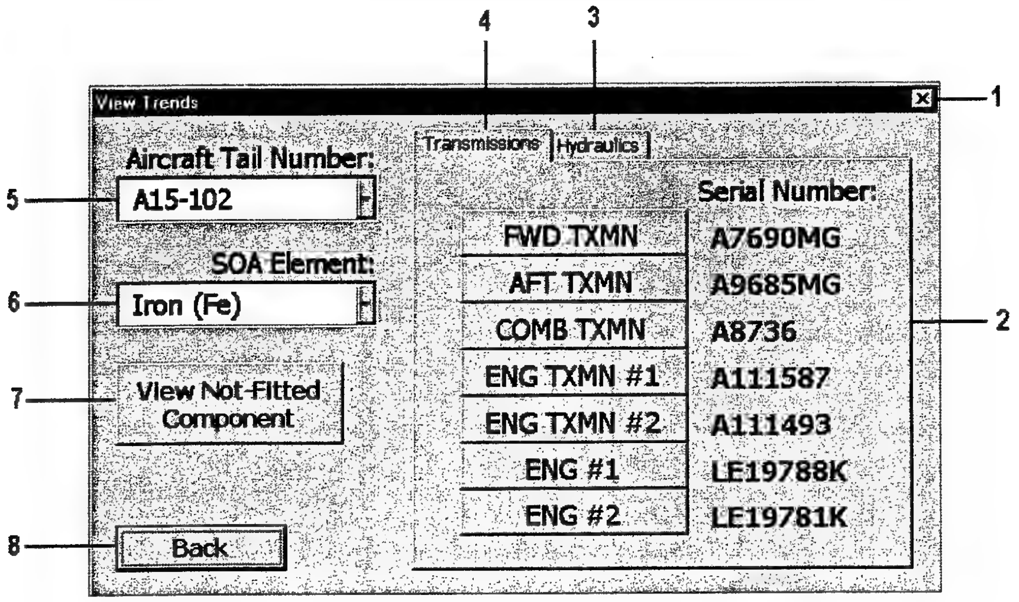
```
Sub SetControls()
    ActiveCell.Offset(0, 1).Select
    ActiveCell.NumberFormat = "@"
    fmHydraulics.sample_date.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.hours.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.sample_no.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.water.ControlSource = ActiveCell.Address

    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Fe.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Cu.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Mg.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Cr.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Al.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Ag.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Sn.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Ni.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Ti.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.Si.ControlSource = ActiveCell.Address

    ActiveCell.Offset(0, 1).Select
    fmHydraulics.viscosity.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    fmHydraulics.code.ControlSource = ActiveCell.Address
End Sub
```

NOTES:
This routine links the appropriate data in the worksheet to the textboxes displayed on the form (2, 4, 6, 7, 11, 12, 13). It is required when the scroll forward and scroll back features are used.

```
Sub reformat_dates()
    Dim the_date As Date
    Dim tmp_day As String
    Dim tmp_month As String
    Dim tmp_year As String
    Dim tmp_date As String

    If ActiveCell.Value <> Empty Then
        the_date = ActiveCell.Value
        tmp_day = Day(the_date)
        tmp_month = Month(the_date)
        tmp_year = Year(the_date)

        ' Reformat to give day-month-year
        Select Case tmp_month
            Case "1"
                tmp_month = "Jan"
            Case "2"
                tmp_month = "Feb"
            Case "3"
                tmp_month = "Mar"
            Case "4"
                tmp_month = "Apr"
            Case "5"
                tmp_month = "May"
            Case "6"
                tmp_month = "Jun"
            Case "7"
                tmp_month = "Jul"
            Case "8"
                tmp_month = "Aug"
            Case "9"
                tmp_month = "Sep"
            Case "10"
                tmp_month = "Oct"
            Case "11"
                tmp_month = "Nov"
            Case "12"
                tmp_month = "Dec"
        End Select

        tmp_date = tmp_day & "/" & tmp_month & "/" & tmp_year
        ActiveCell.NumberFormat = "@"
        ActiveCell.FormulaR1C1 = tmp_date
        ActiveCell.NumberFormat = "dd/mm/yy"
        ActiveCell.NumberFormat = "@"
    End If
End Sub
```

NOTES:
Used to reformat the date entered, forcing it to stay in the Australian format: (day/month/year).

A.5. View trends form: fmTrends



```
Dim OIL_flag As Boolean
Dim cur_plot_sheet As String
Const N_AC As Integer = 6 'number of aircraft in fleet
Const remove_str As String = "REMOVED"
```

NOTES:
Variables local to the form fmTransmission

```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
'    Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub back_Click()
    Unload fmTrends
    fmAircraft.Show
End Sub
```

NOTES:
Unloads the *View Trends* form and returns to the main form: fmAircraft.

```
Private Sub tail_no_Change()
    set_captions
End Sub
```

NOTES:
Resets the component serial numbers displayed on the *Transmissions* page when a new aircraft is selected.

```
Sub Initialize()
    Sheets("Aircraft").Activate
    ActiveSheet.Range("A2").Select

    tail_no.AddItem "All Aircraft"
    While (ActiveCell.Value <> "")
        tail_no.AddItem ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
    Wend
    tail_no.ListIndex = fmAircraft.tail_no.ListIndex + 1

    SOA_element.AddItem "Iron (Fe)"
    SOA_element.AddItem "Copper (Cu)"
    SOA_element.AddItem "Magnesium (Mg)"
    SOA_element.AddItem "Chromium (Cr)"
    SOA_element.AddItem "Aluminium (Al)"
    SOA_element.AddItem "Silver (Ag)"
    SOA_element.AddItem "Tin (Sn)"
    SOA_element.AddItem "Nickel (Ni)"
    SOA_element.AddItem "Titanium (Ti)"
    SOA_element.AddItem "Silicon (Si)"
    SOA_element.ListIndex = 0

    set_captions
    MultiPage1_Change
End Sub
```

NOTES:
Populates the aircraft tail number selection list-box (5) and sets the currently selected aircraft to the same as the main form, fmAircraft. Populates the SOA element list-box (6) with the elements for which plots are available.

```
Sub set_cur_plot_sheet(CPS As String)
    cur_plot_sheet = CPS
End Sub
```

NOTES:
Resets the form variable *cur_plot_sheet*, which controls which charts are used. The sheet *Plot* is used for single plots, and the sheet *Plots* is used for multiple plots.

```
Private Sub MultiPage1_Change()
    Dim tmp_index As Integer

    If (SOA_element.ListIndex = 10) Then
        tmp_index = 0
    Else
        tmp_index = SOA_element.ListIndex
    End If
    SOA_element.Clear
    Select Case MultiPage1.Value
        Case 0
            SOA_element.AddItem "Iron (Fe)"
            SOA_element.AddItem "Copper (Cu)"
            SOA_element.AddItem "Magnesium (Mg)"
            SOA_element.AddItem "Chromium (Cr)"
            SOA_element.AddItem "Aluminium (Al)"
            SOA_element.AddItem "Silver (Ag)"
            SOA_element.AddItem "Tin (Sn)"
            SOA_element.AddItem "Nickel (Ni)"
            SOA_element.AddItem "Titanium (Ti)"
            SOA_element.AddItem "Silicon (Si)"
        Case 1
            SOA_element.AddItem "Iron (Fe)"
            SOA_element.AddItem "Copper (Cu)"
            SOA_element.AddItem "Magnesium (Mg)"
            SOA_element.AddItem "Chromium (Cr)"
            SOA_element.AddItem "Aluminium (Al)"
            SOA_element.AddItem "Silver (Ag)"
            SOA_element.AddItem "Tin (Sn)"
            SOA_element.AddItem "Nickel (Ni)"
            SOA_element.AddItem "Titanium (Ti)"
            SOA_element.AddItem "Silicon (Si)"
            SOA_element.AddItem "Water"
    End Select
    SOA_element.ListIndex = tmp_index
End Sub
```

NOTES:
Resets the SOA element list when a new tab, either *Hydraulics* (3) or *Transmissions* (4) is selected on the page control (2). This is required as water is tested for in the hydraulic systems, but not in the transmissions.

```
Private Sub aft_txmn_Click()
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("C" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                        … " Aft Txmn. ~ S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 1, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("C", "Aft Transmission ",  …
                                                … the_offset + 6, 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 1, the_offset)
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *AFT TXMN* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub comb_txmn_Click()
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("D" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                      … " Comb. Txmn. ~ S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 2, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("D", "Comb. Transmission ", …
                                               … the_offset + 6, 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 2, the_offset)
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *COMB TXMN* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub eng_txmn1_Click()
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("E" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                  … " Eng. #1 Txmn. - S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 3, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("E", "Engine #1 Transmission ", …
                                                … the_offset + 6, 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 3, the_offset)
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *ENG TXMN #1* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub eng_txmn2_Click()
```

```
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("F" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                     … " Eng. #2 Txmn. ~ S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 3, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("F", "Engine #2 Transmission ", …
                                              … the_offset + 6, 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 3, the_offset)
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *ENG TXMN #2* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub eng1_Click()
    Dim TXMN As String
```

```
        Dim the_offset As Integer
        Dim i As Integer

        Application.ScreenUpdating = False
        the_offset = SOA_element.ListIndex
        Sheets("Aircraft").Activate
        Range("G" & (tail_no.ListIndex + 1)).Select
        TXMN = ActiveCell.Value

        OIL_flag = True
        If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
            cur_plot_sheet = "Plot"
            Sheets(cur_plot_sheet).Visible = True
            Sheets(cur_plot_sheet).Activate
            delete_plots (1)
            Call get_single_plot(TXMN, tail_no.Value & …
                                ... " Eng. #1 ~ S/N: ", the_offset + 6, 4, 2)
            Call get_oil_data(TXMN, 1)
            Call add_warning(cur_plot_sheet, 1, 4, the_offset)
            Call format_plot_titles(1, 18, 14)

            Application.ScreenUpdating = True
            fmTrends.Hide
            ActiveSheet.Select
        ElseIf (tail_no.ListIndex = 0) Then
            cur_plot_sheet = "Plots"
            Sheets(cur_plot_sheet).Visible = True
            Sheets(cur_plot_sheet).Activate
            For i = 1 To 6
                delete_plots (i)
            Next i
            Call get_multiple_plots("G", "Engine #1 ", the_offset + 6, …
                                                                    ... 4, 2)
            For i = 1 To 6
                Call add_warning(cur_plot_sheet, i, 4, the_offset)
            Next i

            Application.ScreenUpdating = True
            fmTrends.Hide
            ActiveSheet.Select
        End If
End Sub
```

NOTES:
Called by the *ENG #1* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub eng2_Click()
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("H" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                            … " Eng. #2 ~ S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 4, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("H", "Engine #2 ", the_offset + 6, …
                                                            … 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 4, the_offset)
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *ENG #2* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots,* followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub fwd_txmn_Click()
    Dim TXMN As String
    Dim the_offset As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    the_offset = SOA_element.ListIndex
    Sheets("Aircraft").Activate
    Range("B" & (tail_no.ListIndex + 1)).Select
    TXMN = ActiveCell.Value

    OIL_flag = True
    If ((TXMN <> remove_str) And (tail_no.ListIndex <> 0)) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(TXMN, tail_no.Value & …
                    … " Fwd. Txmn. ~ S/N: ", the_offset + 6, 4, 2)
        Call get_oil_data(TXMN, 1)
        Call add_warning(cur_plot_sheet, 1, 0, the_offset)
        Call format_plot_titles(1, 18, 14)

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    ElseIf (tail_no.ListIndex = 0) Then
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("B", "Fwd. Transmission ", …
                                        … the_offset + 6, 4, 2)
        For i = 1 To 6
            Call add_warning(cur_plot_sheet, i, 0, the_offset)
            Application.ScreenUpdating = False
        Next i

        Application.ScreenUpdating = True
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

NOTES:
Called by the *FWD TXMN* button on the *Transmissions* tab (4). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions. Oil top-up data and the abnormal/warning limits are then added by separate function calls.

```
Private Sub no1_hyd_sys_Click()
    Dim data_col As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    Select Case SOA_element.ListIndex
        Case Is <= 9
            data_col = SOA_element.ListIndex + 6
        Case Is = 10
            data_col = 5
    End Select

    OIL_flag = False
    If (tail_no.ListIndex <> 0) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(tail_no.Value, …
                                  … "No. 1 Hyd. System ~ ", data_col, 2, 3)
        Call format_plot_titles(1, 18, 14)
    Else
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("", "No. 1 Hyd. System ", data_col, …
                                                                    … 2, 3)
    End If
    Application.ScreenUpdating = True
    fmTrends.Hide
End Sub
```

NOTES:
Called by the *#1 HYD SYS* button on the *Hydraulics* tab (3). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots,* followed by a call to one of the plotting functions.

```
Private Sub no2_hyd_sys_Click()
    Dim data_col As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    Select Case SOA_element.ListIndex
        Case Is <= 9
            data_col = SOA_element.ListIndex + 23
        Case Is = 10
            data_col = 22
    End Select

    OIL_flag = False
    If (tail_no.ListIndex <> 0) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(tail_no.Value, "No. 2 Hyd. System ~ ", …
                                                    … data_col, 19, 20)
        Call format_plot_titles(1, 18, 14)
    Else
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("", "No. 2 Hyd. System ", data_col, …
                                                              … 19, 20)
    End If
    Application.ScreenUpdating = True
    fmTrends.Hide
End Sub
```

NOTES:
Called by the *#2 HYD SYS* button on the *Hydraulics* tab (3). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions.

```
Private Sub utility_hyd_Click()
    Dim data_col As Integer
    Dim i As Integer

    Application.ScreenUpdating = False
    Select Case SOA_element.ListIndex
        Case Is <= 9
            data_col = SOA_element.ListIndex + 40
        Case Is = 10
            data_col = 39
    End Select

    OIL_flag = False
    If (tail_no.ListIndex <> 0) Then
        cur_plot_sheet = "Plot"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        delete_plots (1)
        Call get_single_plot(tail_no.Value, …
                         … "Utility Hyd. System ~ ", data_col, 36, 37)
        Call format_plot_titles(1, 18, 14)
    Else
        cur_plot_sheet = "Plots"
        Sheets(cur_plot_sheet).Visible = True
        Sheets(cur_plot_sheet).Activate
        For i = 1 To 6
            delete_plots (i)
        Next i
        Call get_multiple_plots("", "Utility Hyd. System ", …
                                                  … data_col, 36, 37)
    End If
    Application.ScreenUpdating = True
    fmTrends.Hide
End Sub
```

NOTES:
Called by the *UTILITY HYD* button on the *Hydraulics* tab (3). Either a single plot will be produced, or multiple plots (one for each aircraft) will be produced depending on the selection of aircraft tail number (5). Existing plot series are removed by *delete_plots*, followed by a call to one of the plotting functions.

```
Sub set_captions()
    Dim AC_row As Integer
    Application.ScreenUpdating = False

    AC_row = tail_no.ListIndex + 1   ' AC data starts at row 2
                                     ' (ListIndex 0 is "All Aircraft")
    If (tail_no.ListIndex = 0) Then
        fwd_txmn_sn.caption = "All Aircraft"
        aft_txmn_sn.caption = "All Aircraft"
        comb_txmn_sn.caption = "All Aircraft"
        eng_txmn1_sn.caption = "All Aircraft"
        eng_txmn2_sn.caption = "All Aircraft"
        eng1_sn.caption = "All Aircraft"
        eng2_sn.caption = "All Aircraft"
    Else
        Sheets("Aircraft").Activate
        Range("B" & AC_row).Select   ' Fwd Transmission is in column B
        fwd_txmn_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        aft_txmn_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        comb_txmn_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        eng_txmn1_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        eng_txmn2_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        eng1_sn.caption = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        eng2_sn.caption = ActiveCell.Value
    End If
End Sub
```

NOTES:
This function sets the captions displaying the component serial numbers on the *Transmissions* tab (4) of the page control (2). This function is called any time a new aircraft tail number (5) is selected.

```
Sub get_plot(the_plot_sheet As String, plot_index As Integer, …
… the_title As String, y_title As String, refX As String, …
… refY As String)
    Dim series_index As Integer

    Application.ScreenUpdating = False
    Sheets(the_plot_sheet).Visible = True
    Sheets(the_plot_sheet).Activate
    Worksheets(the_plot_sheet).ChartObjects(plot_index).Activate

    ActiveChart.SeriesCollection.NewSeries
    series_index = ActiveChart.SeriesCollection.Count
    With ActiveChart
        .ChartTitle.Characters.Text = the_title
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Text = y_title
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Text = "Equipment …
                                                          … Hours"
        .SeriesCollection(series_index).XValues = refX
        .SeriesCollection(series_index).Values = refY
    End With

    With ActiveChart.Axes(xlValue)
        .MinimumScaleIsAuto = True
        .MaximumScaleIsAuto = True
        .MinorUnitIsAuto = True
        .MajorUnitIsAuto = True
    End With
    With ActiveChart.Axes(xlCategory)
        .MinimumScaleIsAuto = True
        .MaximumScaleIsAuto = True
        .MinorUnitIsAuto = True
        .MajorUnitIsAuto = True
    End With

    ActiveChart.PlotArea.Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With

    ActiveChart.PlotArea.Select
    Selection.Top = 40
    Selection.Width = 550
    Selection.Left = 40

    Selection.Interior.ColorIndex = xlNone
    ActiveChart.Deselect
    ActiveSheet.Select
End Sub
```

NOTES:
This function creates one new plot series on a particular plot (*plot_index*) on the worksheet specified (*the_plot_sheet*). The strings *RefX* and *RefY* point to the cells that contain the data to be plotted.

```
Private Sub not_fitted_Click()
    fmTrends.Hide
    Load fmNotFitted
    fmNotFitted.Initialise
    fmNotFitted.Show
End Sub
```

NOTES:
Called by the *View Not-Fitted Component* button (7). Loads and initialises the form fmNotFitted. This form enables the data from components not currently fitted in any aircraft to plotted.

```
Sub get_single_plot(TXMN As String, TX_type As String, …
… the_col As Integer, date_col As Integer, hours_col As Integer)
    Dim refX As String
    Dim refY As String
    Dim the_title As String
    Dim y_title As String
    Dim SOAP_row As Integer

    Application.ScreenUpdating = False
    Sheets(TXMN).Visible = True
    Sheets(TXMN).Select

    SOAP_row = 5     ' SOAP Data starts on row 5
    Sheets(TXMN).Activate
    ActiveSheet.Range("A" & SOAP_row).Select

    While (ActiveCell.Value <> "")
        SOAP_row = SOAP_row + 1
        ActiveCell.Offset(1, 0).Select
    Wend
    SOAP_row = SOAP_row - 1

    Call reformat_dates(SOAP_row, date_col)

    Range("A4").Select
    ActiveCell.Offset(0, the_col - 1).Select
    the_title = ActiveCell.Value & " SOA Trend ~ " & TX_type & TXMN
    y_title = ActiveCell.Value & " [PPM]"
    refX = "='" & TXMN & "'!R5C" & hours_col & ":R" & …
                                        … SOAP_row & "C" & hours_col
    refY = "='" & TXMN & "'!R5C" & the_col & ":R" & …
                                        … SOAP_row & "C" & the_col
    Sheets(TXMN).Visible = False
    Call get_plot("Plot", 1, the_title, y_title, refX, refY)

    Sheets("Plot").Activate
    Range("A1").Select
    ActiveWindow.Zoom = 75
    Range("A1").Select
    ActiveSheet.ChartObjects(1).Activate     ' Removes focus...
    ActiveChart.Deselect
    ActiveSheet.Select
End Sub
```

NOTES:
This function is called when only one plot is required. The transmission serial number, *TXMN* is used to refer to the sheet where the data is stored. The transmission type *TX_type* is used to create the chart title. The plot data is referred to by the columns *date_col* and *hours_col* which are integers (eg: 1=A, 2=B etc.)

```
Sub get_multiple_plots(TX_col As String, TX_type As String, …
     … the_col As Integer, date_col As Integer, hours_col As Integer)
    Dim refX As String
    Dim refY As String
    Dim the_title As String
    Dim y_title As String
    Dim SOAP_row As Integer
    Dim i As Integer
    Dim TXMN As String
    Dim AC As String
    Dim element As String

    Application.ScreenUpdating = False
    i = 1

    While (i <= N_AC)
        ' Get the aircraft -> transmission
        Sheets("Aircraft").Activate
        Range("A" & (i + 1)).Select
        AC = ActiveCell.Value

        If (TX_col <> "") Then
            Range(TX_col & (i + 1)).Select
            TXMN = ActiveCell.Value
        Else
            TXMN = AC
        End If

        If (TXMN <> remove_str) Then
            SOAP_row = 5     ' SOAP Data starts on row 5
            Sheets(TXMN).Visible = True
            Sheets(TXMN).Select
            Sheets(TXMN).Activate
            ActiveSheet.Range("A" & SOAP_row).Select
            While (ActiveCell.Value <> "")
                SOAP_row = SOAP_row + 1
                ActiveCell.Offset(1, 0).Select
            Wend
            SOAP_row = SOAP_row - 1
            Call reformat_dates(SOAP_row, date_col)

            Range("A4").Select
            ActiveCell.Offset(0, the_col - 1).Select

            the_title = ActiveCell.Value & …
                    … " SOA Trend ~ Aircraft: " & AC & vbCr & TX_type
            If (TX_col <> "") Then
                the_title = the_title & " - S/N: " & TXMN
            End If
            element = ActiveCell.Value
            y_title = element & " [PPM]"
            refX = "='" & TXMN & "'!R5C" & hours_col & ":R" & …
                                          … SOAP_row & "C" & hours_col
            refY = "='" & TXMN & "'!R5C" & the_col & ":R" & …
                                          … SOAP_row & "C" & the_col

            Sheets(TXMN).Visible = False
            Call get_plot("Plots", i, the_title, y_title, refX, refY)
            Application.ScreenUpdating = False

            If (OIL_flag) Then
                Call get_oil_data(TXMN, i)
            End If

            Application.ScreenUpdating = False
            Call format_plot_titles(i, 12, 10)
            Application.ScreenUpdating = False
        End If
        i = i + 1
    Wend

    Sheets("Plots").Activate
    Range("A1").Select
    ActiveWindow.Zoom = 32
    Range("A1").Select
    ActiveCell.Value = element & " SOA Trend ~ All Aircraft ~ " …
                                                      … & TX_type
    ActiveSheet.ChartObjects(1).Activate     ' Removes focus...
    ActiveChart.Deselect
    ActiveSheet.Select
End Sub
```

NOTES:
This function is called when a plot for each aircraft is required. The required transmission serial number from each aircraft is read off the *Aircraft* sheet, *TX_col* controls which type of transmission this is. The plot data is referred to by the columns *date_col* and *hours_col* which are integers (eg: 1=A, 2=B etc.)

```
Sub get_oil_data(TXMN As String, chart_index As Integer)
    Dim MAX_row As Integer
    Dim the_hours As Double
    Dim oil_add As String
    Dim i As Integer

    Application.ScreenUpdating = False

    MAX_row = 5     ' SOAP Data starts on row 5
    Sheets(TXMN).Visible = True
    Sheets(TXMN).Select
    Sheets(TXMN).Activate
    ActiveSheet.Range("A" & MAX_row).Select
    While (ActiveCell.Value <> "")
        MAX_row = MAX_row + 1
        ActiveCell.Offset(1, 0).Select
    Wend
    MAX_row = MAX_row - 1

    Sheets("tmp").Visible = True
    Sheets("tmp").Activate
    Columns(chart_index * 2 - 1).Select
    Selection.ClearContents
    Columns(chart_index * 2).Select
    Selection.ClearContents
    For i = 1 To MAX_row - 5
        Sheets(TXMN).Activate
        ActiveSheet.Range("B" & 5 + (i)).Select ' Hours column
        the_hours = ActiveCell.Value
        ActiveSheet.Range("E" & 5 + (i)).Select ' Oil Added column
        oil_add = ActiveCell.Value
        Sheets("tmp").Activate
        ActiveSheet.Range("A" & 1 + (i - 1) * 3).Select
        ActiveCell.Offset(0, (chart_index - 1) * 2).Select
        ActiveCell.Value = the_hours
        ActiveCell.Offset(1, 0) = the_hours
        ActiveCell.Offset(2, 0) = the_hours
        ActiveCell.Offset(0, 1) = -1
        ActiveCell.Offset(1, 1) = oil_add
        ActiveCell.Offset(2, 1) = -1
    Next i
    Sheets("tmp").Visible = False
    Sheets(TXMN).Visible = False
    Call add_oil_plot(MAX_row, chart_index)
End Sub
```

NOTES:
To display the oil added data as a bar chart overlaid on the line graph of the SOAP data, it must first be reformatted. This is done on a temporary sheet called *"tmp"*. Once the data has been written to the temporary sheet, the function *add_oil_plot* draws the plot onto the selected chart.

```
Sub add_oil_plot(MAX_row As Integer, chart_index As Integer)
    Dim n_series As Integer
    Dim C1 As String
    Dim C2 As String

    Application.ScreenUpdating = False

    Sheets("tmp").Activate
    Range("A1").Select
    ActiveCell.Offset(0, (chart_index - 1) * 2).Select
    C1 = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    C2 = ActiveCell.Address
    Range(C1 & ":" & C2).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.Copy

    Sheets(cur_plot_sheet).Select
    ActiveSheet.ChartObjects(chart_index).Activate
    n_series = ActiveChart.SeriesCollection.Count

    If (MAX_row > 5) Then ' Must be more than one datapoint?
        ActiveChart.SeriesCollection.Paste Rowcol:=xlColumns, …
          … SeriesLabels:=False, CategoryLabels:=True, …
          … Replace:=False, NewSeries:=True
        ActiveChart.SeriesCollection(n_series + 1).Select
        ActiveChart.SeriesCollection(n_series + 1).AxisGroup = 2
        Application.CutCopyMode = False
        With Selection.Border
            .ColorIndex = 57
            .Weight = xlMedium
            .LineStyle = xlContinuous
        End With
        With Selection
            .MarkerStyle = xlNone
            .Smooth = False
        End With
        With ActiveChart
            .Axes(xlValue, xlSecondary).HasTitle = True
            .Axes(xlValue, xlSecondary).AxisTitle.Characters.Text = …
                                                   … "Oil Added [Qts.]"
        End With
        ActiveChart.Axes(xlValue, xlSecondary).AxisTitle.Select
        With Selection.Font
            .Size = 16
        End With
        ActiveChart.Axes(xlValue, xlSecondary).Select
        With Selection.TickLabels.Font
            .Size = 14
        End With
        ActiveChart.Axes(xlValue, xlSecondary).Select
        With ActiveChart.Axes(xlValue, xlSecondary)
            .MinimumScale = 0
        End With
        Selection.TickLabels.NumberFormat = "0.0"
    Else
        ActiveChart.SeriesCollection.NewSeries
        ActiveChart.SeriesCollection(n_series + 1).Select
        ActiveChart.SeriesCollection(2).Select
        With Selection.Border
            .LineStyle = xlNone
        End With
        With Selection
            .MarkerStyle = xlNone
        End With
        ActiveChart.SeriesCollection(n_series + 1).AxisGroup = 2
        With ActiveChart
            .Axes(xlValue, xlSecondary).HasTitle = True
            .Axes(xlValue, xlSecondary).AxisTitle.Characters.Text = …
                                                    … "Oil Added [Qts.]"
        End With
    End If

    ActiveChart.Deselect
    ActiveSheet.Select
End Sub
```

NOTES:
This function reads the formatted data off the temporary sheet, *"tmp"*, and adds it as a new series to the plot referred to by *chart_index*. The oil-added data is plotted against a secondary scale displayed on the right hand side of the chart.

```
Sub delete_plots(chart_index As Integer)
    Application.ScreenUpdating = False
    ActiveSheet.ChartObjects(chart_index).Activate

    While (ActiveChart.SeriesCollection.Count > 0)
        ActiveChart.PlotArea.Select
        ActiveChart.SeriesCollection(1).Select
        Selection.Delete
    Wend
End Sub
```

NOTES:
This function erases any series that exist on a given chart.

```
Sub format_plot_titles(plot_index As Integer, big As Integer, …
                                                  … small As Integer)
    Application.ScreenUpdating = False

    ActiveSheet.ChartObjects(plot_index).Activate
    ActiveChart.ChartTitle.Select
    Selection.AutoScaleFont = True
    With Selection.Font
        .Name = "Arial"
        .Size = big
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .Background = xlAutomatic
    End With
    ActiveChart.Axes(xlValue).AxisTitle.Select
    Selection.AutoScaleFont = True
    With Selection.Font
        .Name = "Arial"
        .Size = big
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .Background = xlAutomatic
    End With
    If (OIL_flag) Then
        ActiveChart.Axes(xlValue, xlSecondary).AxisTitle.Select
        Selection.AutoScaleFont = True
        With Selection.Font
            .Name = "Arial"
            .Size = big
            .Strikethrough = False
            .Superscript = False
            .Subscript = False
            .OutlineFont = False
            .Shadow = False
            .Underline = xlUnderlineStyleNone
            .ColorIndex = xlAutomatic
            .Background = xlAutomatic
        End With
    End If
    ActiveChart.Axes(xlValue).Select
    Selection.TickLabels.AutoScaleFont = True
    With Selection.TickLabels.Font
        .Name = "Arial"
        .Size = small
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .Background = xlAutomatic
    End With
    ActiveChart.Axes(xlCategory).Select
    Selection.TickLabels.AutoScaleFont = True
    With Selection.TickLabels.Font
        .Name = "Arial"
        .Size = small
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .Background = xlAutomatic
    End With
    If (OIL_flag) Then
        ActiveChart.Axes(xlValue, xlSecondary).Select
        Selection.TickLabels.AutoScaleFont = True
        With Selection.TickLabels.Font
            .Name = "Arial"
            .Size = small
            .Strikethrough = False
            .Superscript = False
            .Subscript = False
            .OutlineFont = False
            .Shadow = False
            .Underline = xlUnderlineStyleNone
            .ColorIndex = xlAutomatic
            .Background = xlAutomatic
        End With
    End If
    ActiveChart.Deselect
    ActiveSheet.Select
End Sub
```

NOTES:
This function is used to set the font size and styles on the chart referred to by *plot_index*.

```
Sub reformat_dates(MAX_row As Integer, date_col As Integer)
    Dim i As Integer
    Dim col As String
    Dim the_date As Date
    Dim tmp_day As String
    Dim tmp_month As String
    Dim tmp_year As String
    Dim tmp_date As String

    Application.ScreenUpdating = False

    ActiveSheet.Range("A1").Select
    ActiveCell.Offset(4, date_col - 1).Select
    For i = 5 To MAX_row
        If ActiveCell.Value <> Empty Then
            the_date = ActiveCell.Value
            tmp_day = Day(the_date)
            tmp_month = Month(the_date)
            tmp_year = Year(the_date)

            ' Reformat to give day-month-year
            Select Case tmp_month
                Case "1"
                    tmp_month = "Jan"
                Case "2"
                    tmp_month = "Feb"
                Case "3"
                    tmp_month = "Mar"
                Case "4"
                    tmp_month = "Apr"
                Case "5"
                    tmp_month = "May"
                Case "6"
                    tmp_month = "Jun"
                Case "7"
                    tmp_month = "Jul"
                Case "8"
                    tmp_month = "Aug"
                Case "9"
                    tmp_month = "Sep"
                Case "10"
                    tmp_month = "Oct"
                Case "11"
                    tmp_month = "Nov"
                Case "12"
                    tmp_month = "Dec"
            End Select
            tmp_date = tmp_day & "/" & tmp_month & "/" & tmp_year
            ActiveCell.NumberFormat = "dd/mmm/yy"
            ActiveCell.FormulaR1C1 = tmp_date
        End If
        ActiveCell.Offset(1, 0).Select
    Next i
```

```
End Sub
```

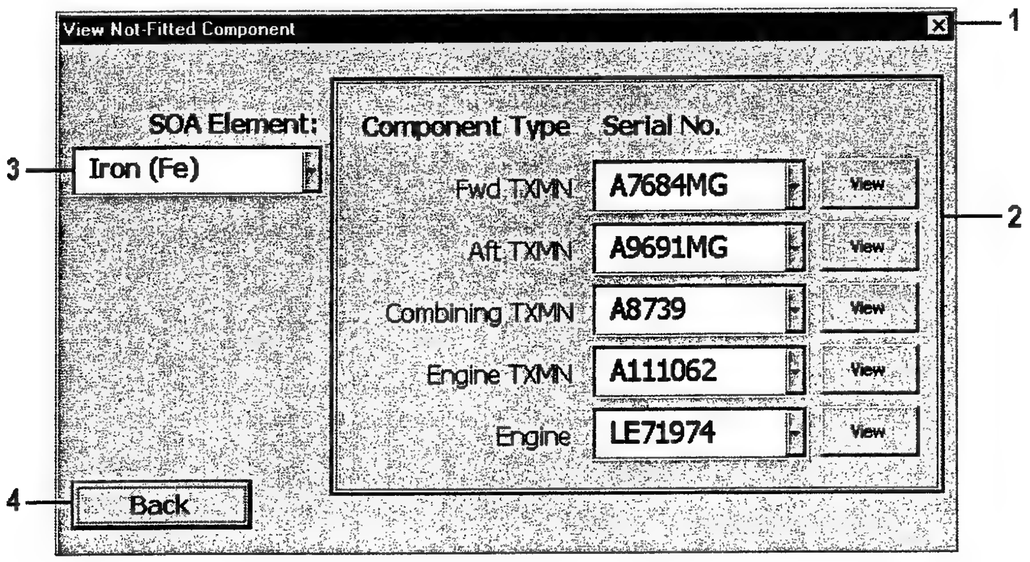
NOTES:
Used to reformat the date entered, forcing it to stay in the Australian format: (day/month/year).

```
Sub add_warning(the_sheet As String, chart_index As Integer, …
                                … TX_type As Integer, ELtype As Integer)
    Dim n_series As Integer
    Dim Xmin_scale As Double
    Dim Xmax_scale As Double
    Dim Ymin_scale As Double
    Dim Ymax_scale As Double
    Dim abnormal As Double
    Dim marginal As Double

    Application.ScreenUpdating = False

    ' Get Levels...
    Sheets("Levels").Visible = True
    Sheets("Levels").Select
    Range("B3").Select        ' Abnormal level data starts on row 3
    ActiveCell.Offset(TX_type, ELtype).Select
    marginal = ActiveCell.Value
    Range("B14").Select       ' Warning level data starts on row 14
    ActiveCell.Offset(TX_type, ELtype).Select
    abnormal = ActiveCell.Value

    ' Activate the chart
    Sheets(the_sheet).Select
    ActiveSheet.ChartObjects(chart_index).Activate

    ' Do nothing if chart blank...
    If (ActiveChart.SeriesCollection.Count > 0) Then
        ' Get Automatic chart range...
        ActiveChart.Axes(xlValue).Select
        With ActiveChart.Axes(xlValue)
            Ymin_scale = .MinimumScale
            Ymax_scale = .MaximumScale
        End With
        With ActiveChart.Axes(xlCategory)
            Xmin_scale = .MinimumScale
            Xmax_scale = .MaximumScale
        End With
        n_series = ActiveChart.SeriesCollection.Count

        ' ABNORMAL
        ActiveChart.SeriesCollection.NewSeries
        ActiveChart.SeriesCollection(n_series + 1).XValues = …
                          … "={" & Xmin_scale & "," & Xmax_scale & "}"
        ActiveChart.SeriesCollection(n_series + 1).Values = …
                          … "={" & marginal & "," & marginal & "}"
        ActiveChart.SeriesCollection(n_series + 1).Select
        With Selection.Border
```

```
                .ColorIndex = 45
                .Weight = xlMedium
                .LineStyle = xlDash
            End With
            With Selection
                .MarkerStyle = xlNone
            End With

            ' WARNING
            ActiveChart.SeriesCollection.NewSeries
            ActiveChart.SeriesCollection(n_series + 2).XValues = …
                              … "={" & Xmin_scale & "," & Xmax_scale & "}"
            ActiveChart.SeriesCollection(n_series + 2).Values = …
                              … "={" & abnormal & "," & abnormal & "}"
            ActiveChart.SeriesCollection(n_series + 2).Select
            With Selection.Border
                .ColorIndex = 3
                .Weight = xlMedium
                .LineStyle = xlDash
            End With
            With Selection
                .MarkerStyle = xlNone
            End With

            ' Reset scaling to auto-selected values from before...
            With ActiveChart.Axes(xlValue)
                .MinimumScale = Ymin_scale
                .MaximumScale = Ymax_scale
            End With
            With ActiveChart.Axes(xlCategory)
                .MinimumScale = Xmin_scale
                .MaximumScale = Xmax_scale
            End With
        End If

        Sheets("Levels").Visible = False
    End Sub
```

NOTES:
This function adds a yellow or red dashed horizontal line indicating the MARGINAL and ABNORMAL levels respectively.

A.6. View not-fitted component form: fmNotFitted



```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub back_Click()
    Unload fmNotFitted
    fmTrends.Show
End Sub
```

NOTES:
Called by the *Back* button (4). Unloads the view not-fitted components form (fmNotFitted) and returns to the view trends form (fmTrends).

```
Private Sub aft_plot_Click()
    get_plot (aft_txmn.Value)
End Sub

Private Sub comb_plot_Click()
    get_plot (comb_txmn.Value)
End Sub

Private Sub eng_plot_Click()
    get_plot (eng.Value)
End Sub

Private Sub eng_tx_plot_Click()
    get_plot (eng_txmn.Value)
End Sub

Private Sub fwd_plot_Click()
    get_plot (fwd_txmn.Value)
End Sub
```

NOTES:
These five functions are called by the *View* buttons (2) next to the listing of each of the transmission types (2). It passes the current transmission serial number to the *get_plot* function.

```
Sub get_plot(TXMN As String)
    Dim the_offset As Integer
    Dim i As Integer

    If (TXMN <> Empty) Then
        Application.ScreenUpdating = False
        the_offset = SOA_element.ListIndex
        Sheets("Plot").Visible = True
        Sheets("Plot").Activate
        fmTrends.delete_plots (1)
        fmTrends.set_cur_plot_sheet ("Plot")
        Call fmTrends.get_single_plot(TXMN, " Fwd. Txmn. - S/N: ", …
                                          … the_offset + 6, 4, 2)
        Call fmTrends.get_oil_data(TXMN, 1)
        Call fmTrends.add_warning("Plot", 1, 0, the_offset)
        Call fmTrends.format_plot_titles(1, 18, 14)
        Application.ScreenUpdating = True
        Unload fmNotFitted
        fmTrends.Hide
        ActiveSheet.Select
    End If
End Sub
```

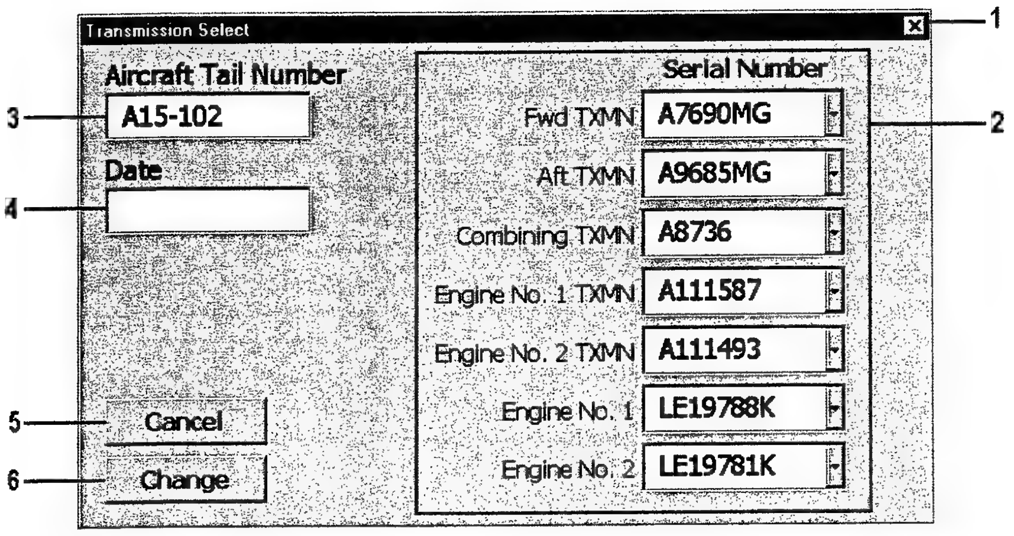
NOTES:
Given the transmission serial number, this function calls the plotting functions from the form *fmTrends*.

```
Sub Initialise()
    Dim cur_txmn As String
    Dim index As Integer
    Dim add_flag As Boolean
    Dim tx_loc As String

    Application.ScreenUpdating = False

    SOA_element.AddItem "Iron (Fe)"
    SOA_element.AddItem "Copper (Cu)"
    SOA_element.AddItem "Magnesium (Mg)"
    SOA_element.AddItem "Chromium (Cr)"
    SOA_element.AddItem "Aluminium (Al)"
    SOA_element.AddItem "Silver (Ag)"
    SOA_element.AddItem "Tin (Sn)"
    SOA_element.AddItem "Nickel (Ni)"
    SOA_element.AddItem "Titanium (Ti)"
    SOA_element.AddItem "Silicon (Si)"
    SOA_element.ListIndex = 0

    Sheets("Transmissions").Visible = True
    Sheets("Aircraft").Visible = True

    'FWD TXMN
    Sheets("Transmissions").Activate
    ActiveSheet.Range("A2").Select
    While (ActiveCell.Value <> "")
        Sheets("Transmissions").Activate
        cur_txmn = ActiveCell.Value
        tx_loc = ActiveCell.Address

        Sheets("Aircraft").Activate
        ActiveSheet.Range("B2").Select
        add_flag = True
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        If (add_flag = True) Then
            fwd_txmn.AddItem cur_txmn
        End If
        Sheets("Transmissions").Activate
        ActiveSheet.Range(tx_loc).Select
        ActiveCell.Offset(1, 0).Select
    Wend

    'AFT TXMN
    Sheets("Transmissions").Activate
    ActiveSheet.Range("B2").Select
    While (ActiveCell.Value <> "")
        Sheets("Transmissions").Activate
        cur_txmn = ActiveCell.Value
        tx_loc = ActiveCell.Address

        Sheets("Aircraft").Activate
        ActiveSheet.Range("C2").Select
        add_flag = True
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        If (add_flag = True) Then
            aft_txmn.AddItem cur_txmn
        End If
        Sheets("Transmissions").Activate
        ActiveSheet.Range(tx_loc).Select
        ActiveCell.Offset(1, 0).Select
    Wend

    'COMB TXMN
    Sheets("Transmissions").Activate
    ActiveSheet.Range("C2").Select
    While (ActiveCell.Value <> "")
        Sheets("Transmissions").Activate
        cur_txmn = ActiveCell.Value
        tx_loc = ActiveCell.Address

        Sheets("Aircraft").Activate
        ActiveSheet.Range("D2").Select
        add_flag = True
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        If (add_flag = True) Then
            comb_txmn.AddItem cur_txmn
        End If
        Sheets("Transmissions").Activate
        ActiveSheet.Range(tx_loc).Select
        ActiveCell.Offset(1, 0).Select
    Wend

    'ENG TXMN
    Sheets("Transmissions").Activate
    ActiveSheet.Range("D2").Select
    ActiveCell.Offset(0, 0).Select
    While (ActiveCell.Value <> "")
        Sheets("Transmissions").Activate
        cur_txmn = ActiveCell.Value
        tx_loc = ActiveCell.Address

        Sheets("Aircraft").Activate
        ActiveSheet.Range("E2").Select
        add_flag = True
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        ActiveSheet.Range("F2").Select
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        If (add_flag = True) Then
            eng_txmn.AddItem cur_txmn
        End If
        Sheets("Transmissions").Activate
        ActiveSheet.Range(tx_loc).Select
        ActiveCell.Offset(1, 0).Select
    Wend

    'ENG
    Sheets("Transmissions").Activate
    ActiveSheet.Range("E2").Select
    ActiveCell.Offset(0, 0).Select
    While (ActiveCell.Value <> "")
        Sheets("Transmissions").Activate
        cur_txmn = ActiveCell.Value
        tx_loc = ActiveCell.Address

        Sheets("Aircraft").Activate
        ActiveSheet.Range("G2").Select
        add_flag = True
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        ActiveSheet.Range("H2").Select
        While (ActiveCell.Value <> "")
            If (ActiveCell.Value = cur_txmn) Then
                add_flag = False
            End If
            ActiveCell.Offset(1, 0).Select
        Wend
        If (add_flag = True) Then
            eng.AddItem cur_txmn
        End If
        Sheets("Transmissions").Activate
        ActiveSheet.Range(tx_loc).Select
        ActiveCell.Offset(1, 0).Select
    Wend
```

```
        If (fwd_txmn.ListCount > 0) Then
            fwd_txmn.ListIndex = 0
        End If
        If (aft_txmn.ListCount > 0) Then
            aft_txmn.ListIndex = 0
        End If
        If (comb_txmn.ListCount > 0) Then
            comb_txmn.ListIndex = 0
        End If
        If (eng_txmn.ListCount > 0) Then
            eng_txmn.ListIndex = 0
        End If
        If (eng.ListCount > 0) Then
            eng.ListIndex = 0
        End If

        Sheets("Transmissions").Visible = False
        Sheets("Aircraft").Visible = False
        Application.ScreenUpdating = True
End Sub
```

NOTES:
For the five transmission types a component is added to the not-fitted list-box if it is on the master list of components on the *Transmissions* page but does not appear in the list of installed transmissions on the *Aircraft* sheet.

A.7. Change transmission component form: fmTransmission



Aircraft Tail Number: A15-102

Date:

Fwd TXMN	A7690MG
Aft TXMN	A9685MG
Combining TXMN	A8736
Engine No. 1 TXMN	A111587
Engine No. 2 TXMN	A111493
Engine No. 1	LE19788K
Engine No. 2	LE19781K

```
Const N_AC As Integer = 6    ' 6 Aircraft in fleet
Const N_TX As Integer = 7    ' 7 Transmissions in Aircraft
Const remove_str As String = "REMOVED"

Dim TXarray(7, 2) As Integer
Dim AC_row As Integer
Dim fwd_txmn_index As Integer
Dim aft_txmn_index As Integer
Dim comb_txmn_index As Integer
Dim eng_txmn1_index As Integer
Dim eng_txmn2_index As Integer
Dim eng1_index As Integer
Dim eng2_index As Integer

Option Compare Text ' That is, "AAA" is equal to "aaa".
```

NOTES:
Variables local to the form fmTransmission

```
Private Sub UserForm_QueryClose _
  (cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub Cancel_Click()
    fwd_txmn.ListIndex = TXarray(0, 1)
    aft_txmn.ListIndex = TXarray(1, 1)
    comb_txmn.ListIndex = TXarray(2, 1)
    eng_txmn1.ListIndex = TXarray(3, 1)
    eng_txmn2.ListIndex = TXarray(4, 1)
    eng1.ListIndex = TXarray(5, 1)
    eng2.ListIndex = TXarray(6, 1)

    Sheets("Aircraft").Visible = True
    Sheets("Aircraft").Activate

    ActiveSheet.Range("Aircraft!B" & AC_row).Select
    ActiveCell.Value = fwd_txmn.Value
    ActiveSheet.Range("Aircraft!C" & AC_row).Select
    ActiveCell.Value = aft_txmn.Value
    ActiveSheet.Range("Aircraft!D" & AC_row).Select
    ActiveCell.Value = comb_txmn.Value
    ActiveSheet.Range("Aircraft!E" & AC_row).Select
    ActiveCell.Value = eng_txmn1.Value
    ActiveSheet.Range("Aircraft!F" & AC_row).Select
    ActiveCell.Value = eng_txmn2.Value
    ActiveSheet.Range("Aircraft!G" & AC_row).Select
    ActiveCell.Value = eng1.Value
    ActiveSheet.Range("Aircraft!H" & AC_row).Select
    ActiveCell.Value = eng2.Value


    Sheets("Aircraft").Visible = False
    Unload fmTransmission
    fmAircraft.Show
End Sub
```

NOTES:
Called when the *Cancel* button (5) is pressed. Exits from the transmission change page without altering the currently installed transmissions in the selected aircraft. This is accomplished by restoring the original state of the aircraft that is saved in TXarray by the *Initialize* function when the form is loaded. Returns to the main form fmAircraft.

```
Sub Initialize()
    Dim index As Integer
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim n_end As Integer
    Dim txmn_object As Control

    Application.ScreenUpdating = False

    TXarray(0, 0) = fwd_txmn.TabIndex
    TXarray(1, 0) = aft_txmn.TabIndex
    TXarray(2, 0) = comb_txmn.TabIndex
    TXarray(3, 0) = eng_txmn1.TabIndex
    TXarray(4, 0) = eng_txmn2.TabIndex
    TXarray(5, 0) = eng1.TabIndex
    TXarray(6, 0) = eng2.TabIndex


    ' AC data starts at row 2
    AC_row = fmAircraft.tail_no.ListIndex + 2
    tail_no.ControlSource = "Aircraft!A" & AC_row

    j = 0
    For Each txmn_object In Controls
        If (j < N_TX) Then
            If txmn_object.TabIndex = TXarray(j, 0) Then

                ' NOTE: Transmission data starts at row 2...
                Sheets("Aircraft").Activate
                ActiveSheet.Range("Aircraft!B" & AC_row).Select

                ' Move to correct transmission type
                ActiveCell.Offset(0, j).Select

                ' Build list of available transmissions
                Select Case j
                    Case Is < 3  '0,1,2
                        k = j
                    Case 3 To 4
                        k = 3
                    Case 5 To 6
                        k = 4
                End Select

                ' Builds menu using all transmissions...
                Sheets("Transmissions").Activate
                txmn_object.AddItem remove_str
                ActiveSheet.Range("A2").Select
                ActiveCell.Offset(0, k).Select

                txmn_object.ListIndex = 0
                While (ActiveCell.Value <> Empty)
                    txmn_object.AddItem ActiveCell.Value
                    ActiveCell.Offset(1, 0).Select
                Wend



        ' Remove from menu those components that are already fitted...
        ' Unless fitted to current AC in that location!
                Sheets("Aircraft").Activate
                ActiveSheet.Range("Aircraft!B" & AC_row).Select
                ActiveCell.Offset(0, j).Select
                While (ActiveCell.Value <> Empty)
                    n_end = txmn_object.ListCount
                    For i = 0 To n_end - 1
            If i < txmn_object.ListCount Then
                txmn_object.ListIndex = i
                If txmn_object.Value = ActiveCell.Value …
                                                    … Then
                    If ActiveCell.Row <> AC_row Then
                        txmn_object.RemoveItem i
                    End If
                    n_end = txmn_object.ListCount
                End If
            End If
        Next i
        ActiveCell.Offset(1, 0).Select
    Wend

    ' Case for engines and eng TX
    If (j = 3 Or j = 5) Then
    ActiveSheet.Range("Aircraft!B" & AC_row).Select
    ActiveCell.Offset(0, j + 1).Select
        While (ActiveCell.Value <> Empty)
            n_end = txmn_object.ListCount
            For i = 0 To n_end - 1
                If i < txmn_object.ListCount Then
                    txmn_object.ListIndex = i
                    If txmn_object.Value = …
                                  … ActiveCell.Value Then
                        txmn_object.RemoveItem i
                        n_end = txmn_object.ListCount
                    End If
                End If
            Next i
            ActiveCell.Offset(1, 0).Select
        Wend
    End If

    ' Case for engines and eng TX
    If (j = 4 Or j = 6) Then
    ActiveSheet.Range("Aircraft!B" & AC_row).Select
    ActiveCell.Offset(0, j - 1).Select
        While (ActiveCell.Value <> Empty)
            n_end = txmn_object.ListCount
            For i = 0 To n_end - 1
                If i < txmn_object.ListCount Then
                    txmn_object.ListIndex = i
                    If txmn_object.Value = …
                                  … ActiveCell.Value Then
                        txmn_object.RemoveItem i
                        n_end = txmn_object.ListCount
                    End If
                End If
            Next i
            ActiveCell.Offset(1, 0).Select
        Wend
    End If
    Sheets("Aircraft").Activate
    ActiveSheet.Range("B" & AC_row).Select
```

```
                ActiveCell.Offset(j, 0).Select
                txmn_object.ListIndex = 0
                n_end = txmn_object.ListCount
                For i = 0 To n_end - 1
                    txmn_object.ListIndex = i
                    If (txmn_object.Value = ActiveCell.Value) Then
                        index = i
                    End If
                Next i
                txmn_object.ListIndex = index

                j = j + 1
            End If
        End If ' j < N_TX
    Next 'txmn_object

    fwd_txmn.ControlSource = "Aircraft!B" & AC_row
    aft_txmn.ControlSource = "Aircraft!C" & AC_row
    comb_txmn.ControlSource = "Aircraft!D" & AC_row
    eng_txmn1.ControlSource = "Aircraft!E" & AC_row
    eng_txmn2.ControlSource = "Aircraft!F" & AC_row
    eng1.ControlSource = "Aircraft!G" & AC_row
    eng2.ControlSource = "Aircraft!H" & AC_row

    fwd_txmn_index = fwd_txmn.ListIndex
    aft_txmn_index = aft_txmn.ListIndex
    comb_txmn_index = comb_txmn.ListIndex
    eng_txmn1_index = eng_txmn1.ListIndex
    eng_txmn2_index = eng_txmn2.ListIndex
    eng1_index = eng1.ListIndex
    eng2_index = eng2.ListIndex

    TXarray(0, 1) = fwd_txmn_index
    TXarray(1, 1) = aft_txmn_index
    TXarray(2, 1) = comb_txmn_index
    TXarray(3, 1) = eng_txmn1_index
    TXarray(4, 1) = eng_txmn2_index
    TXarray(5, 1) = eng1_index
    TXarray(6, 1) = eng2_index

End Sub
```

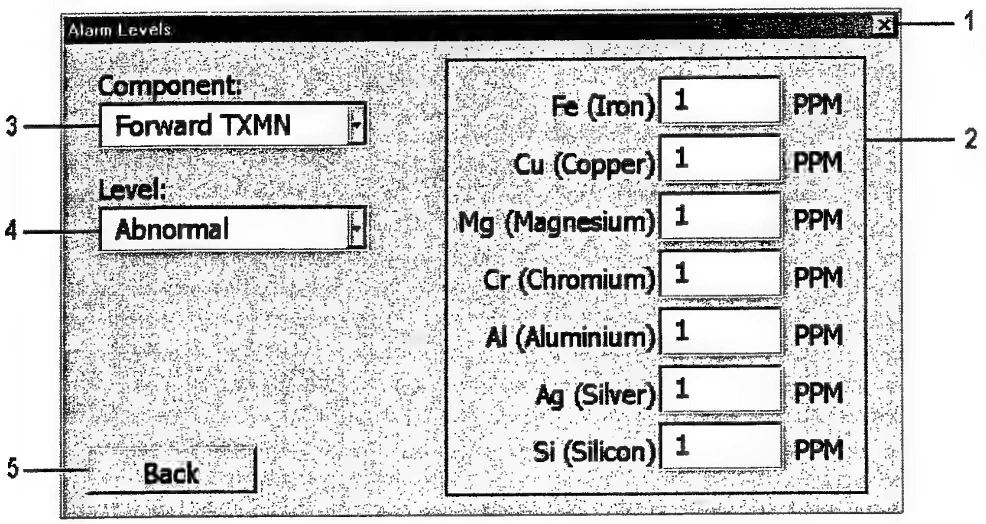
NOTES:
This function populates the list boxes (2) with the available options for transmission components that may be installed in the selected aircraft (3). The available options include those components that are currently not fitted to any aircraft. Each list initially shows the serial number of the component that is currently fitted. Each list-box is bound to the aircraft data page where the currently installed component serial number is stored. Any changes made to the installed component are therefore automatically updated here. An image of the original state of the aircraft is kept in TXarray - this information is used in the event that the operation is cancelled.

```
Private Sub Change_Button_Click()
    Dim TXMN As String
    Dim AC As String
    Dim SOAP_row As Integer
    Dim tmp_index As Integer
    Dim txmn_object As Control
    Dim i As Integer
    Dim response As String
    Dim tmp_day As String
    Dim tmp_month As String
    Dim tmp_year As String
    Dim tmp_date As String
    Dim change_tx As Boolean
    Dim component_hours As Double

    change_tx = False
    If (fwd_txmn_index <> fwd_txmn.ListIndex) Then
        change_tx = True
    End If
    If (aft_txmn_index <> aft_txmn.ListIndex) Then
        change_tx = True
    End If
    If (comb_txmn_index <> comb_txmn.ListIndex) Then
        change_tx = True
    End If
    If (eng_txmn1_index <> eng_txmn1.ListIndex) Then
        change_tx = True
    End If
    If (eng_txmn2_index <> eng_txmn2.ListIndex) Then
        change_tx = True
    End If
    If (eng1_index <> eng1.ListIndex) Then
        change_tx = True
    End If
    If (eng2_index <> eng2.ListIndex) Then
        change_tx = True
    End If

    If (the_date.Value <> Empty) Then

        tmp_day = Day(the_date.Value)
        tmp_month = Month(the_date.Value)
        tmp_year = Year(the_date.Value)

        ' Reformat to give day-month-year
        Select Case tmp_month
            Case "1"
                tmp_month = "Jan"
            Case "2"
                tmp_month = "Feb"
            Case "3"
                tmp_month = "Mar"
            Case "4"
            tmp_month = "Apr"
        Case "5"
            tmp_month = "May"
        Case "6"
            tmp_month = "Jun"
        Case "7"
            tmp_month = "Jul"
        Case "8"
            tmp_month = "Aug"
        Case "9"
            tmp_month = "Sep"
        Case "10"
            tmp_month = "Oct"
        Case "11"
            tmp_month = "Nov"
        Case "12"
            tmp_month = "Dec"
    End Select
    tmp_date = tmp_day & "/" & tmp_month & "/" & tmp_year
    the_date.Value = tmp_date

    ' Get AC identity...
    AC = fmAircraft.tail_no.Text
    For Each txmn_object In Controls
        i = 0
        While i < N_TX

            If txmn_object.TabIndex = TXarray(i, 0) Then
                Sheets("Aircraft").Visible = True
                Sheets("Aircraft").Activate
                ' Get transmission identity...
                tmp_index = txmn_object.ListIndex
                txmn_object.ListIndex = TXarray(i, 1)
                TXMN = txmn_object.Value
                txmn_object.ListIndex = tmp_index
                If (txmn_object.ListIndex <> TXarray(i, 1)) Then
                    If (TXarray(i, 1) <> 0) Then
                        ' Remove existing component
               ' SOAP Data starts on row 5...
                        SOAP_row = 5
                        Sheets(TXMN).Activate
                        ActiveSheet.Range("A" & SOAP_row).Select
                        While (ActiveCell.Value <> "")
                            SOAP_row = SOAP_row + 1
                            ActiveCell.Offset(1, 0).Select
                        Wend
                        ActiveSheet.Range("A" & SOAP_row).Select
                        ActiveCell.Value = "Removed from " & AC
                        ActiveSheet.Range("D" & SOAP_row).Select
                        ActiveCell.Value = the_date.Value
                    End If
                    If (txmn_object.ListIndex <> 0) Then
                        ' Add new component
                        TXMN = txmn_object.Value
                        ' SOAP Data starts on row 5...
                            SOAP_row = 5
                            Sheets(TXMN).Activate
                            ActiveSheet.Range("A" & SOAP_row).Select
                            While (ActiveCell.Value <> "")
                                SOAP_row = SOAP_row + 1
                                ActiveCell.Offset(1, 0).Select
                            Wend
                            ActiveSheet.Range("A" & SOAP_row).Select
                            ActiveCell.Value = "Fitted to " & AC
                            ActiveSheet.Range("D" & SOAP_row).Select
                            ActiveCell.Value = the_date.Value
                        End If
                        If (SOAP_row > 5) Then
                        ' Copy component hours from previous value
                            ActiveSheet.Range("B" & SOAP_row -1). …
                                    … Select
                            component_hours = ActiveCell.Value
                            ActiveSheet.Range("B" & SOAP_row).Select
                            ActiveCell.Value = component_hours
                        End If
                    End If

                End If
                i = i + 1
            Wend
        Next    'txmn_object

        Unload fmTransmission
        fmAircraft.Show
    ElseIf (change_tx = True) Then
        response = MsgBox("Please enter a date for this action", …
                  … vbOKOnly + vbExclamation + vbApplicationModal)
    Else
        Unload fmTransmission
        fmAircraft.Show
    End If
    Sheets("Aircraft").Visible = False
End Sub
```

NOTES:
If a transmission has been changed, and a date for the change has been entered then this function will update the database in three areas:

1. On the *Aircraft* page, the new installed component serial numbers are updated.
2. On the *Removed Components* page, a note is made of when and from which aircraft it was removed.
3. On the *Installed Components* page, a note is made of when and to which aircraft it was installed.

A.8. Set alarm levels form: fmLevels



Component: Forward TXMN

Level: Abnormal

Fe (Iron)	1	PPM
Cu (Copper)	1	PPM
Mg (Magnesium)	1	PPM
Cr (Chromium)	1	PPM
Al (Aluminium)	1	PPM
Ag (Silver)	1	PPM
Si (Silicon)	1	PPM

```
Private Sub UserForm_QueryClose _
  (Cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        Cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub back_Click()
    Unload fmLevels
    fmAircraft.Show
End Sub
```

NOTES:
Called when *Back* button (5) is pressed. Closes the *Alarm Levels* form and returns to the main page - any changes are kept.

```
Private Sub Component_Change()
    get_levels
End Sub
```

NOTES:
Called when the component type (3) is changed. Alarm level data is updated.

```
Private Sub Level_Change()
    get_levels
End Sub
```

NOTES:
Called when the component type (4) is changed. Alarm level data is updated.

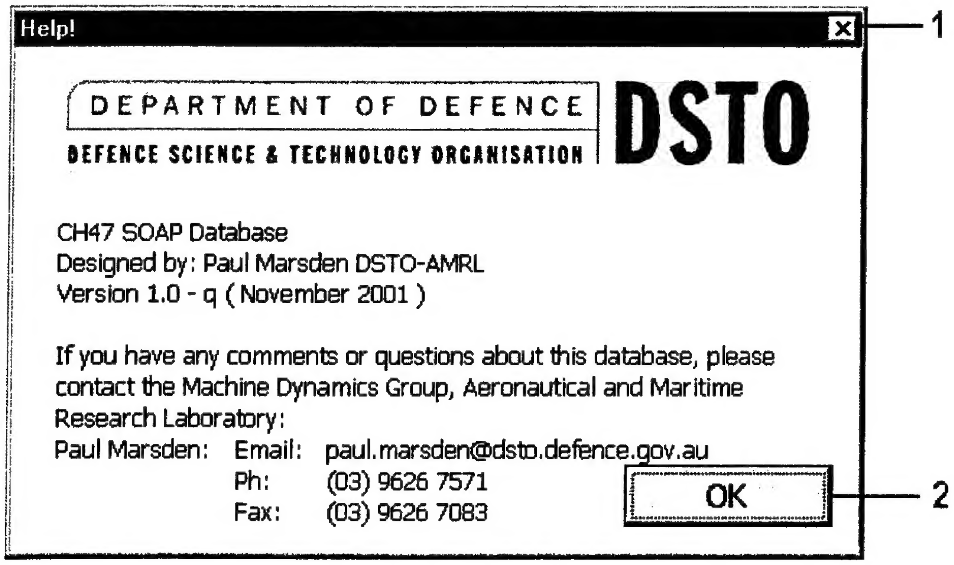
```
Sub Initialize()
    Component.AddItem ("Forward TXMN")
    Component.AddItem ("Aft TXMN")
    Component.AddItem ("Combining TXMN")
    Component.AddItem ("Engine TXMN")
    Component.AddItem ("Engine")
    Component.ListIndex = 0

    Level.AddItem ("Marginal")
    Level.AddItem ("Abnormal")
    Level.ListIndex = 0

    get_levels
End Sub
```

NOTES:
Initialises the list boxes on the form with component names (3) and alarm levels (4). The alarm level values are set by the function *get_levels*.

```
Private Sub get_levels()
    Application.ScreenUpdating = False

    Sheets("Levels").Visible = True
    Sheets("Levels").Activate

    If (Level.ListIndex = 0) Then
        Range("B3").Select      ' Abnormal level data starts on row 3
    ElseIf (Level.ListIndex = 1) Then
        Range("B14").Select     ' Warning level data starts on row 14
    End If
    ActiveCell.Offset(Component.ListIndex, 0).Select

    Fe.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    Cu.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    Mg.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    Cr.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    Al.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 1).Select
    Ag.ControlSource = ActiveCell.Address
    ActiveCell.Offset(0, 4).Select  ' Skip: Tin, Nickel, Titanium
    Si.ControlSource = ActiveCell.Address
    Sheets("Levels").Visible = False
    Application.ScreenUpdating = False
End Sub
```

NOTES:
Sets the alarm level values (2) for the indicated component and alarm level type. The *ControlSource* call links the displayed text directly to the stored data on the sheet *Levels*. There is no error checking - valid numbers must be entered.

A.9. Help form: fmHelp



CH47 SOAP Database
Designed by: Paul Marsden DSTO-AMRL
Version 1.0 - q (November 2001)

If you have any comments or questions about this database, please contact the Machine Dynamics Group, Aeronautical and Maritime Research Laboratory:

Paul Marsden:	Email:	paul.marsden@dsto.defence.gov.au
	Ph:	(03) 9626 7571
	Fax:	(03) 9626 7083

```
Private Sub UserForm_QueryClose _
  (Cancel As Integer, CloseMode As Integer)
'   Prevents use of the Close button
    If CloseMode = vbFormControlMenu Then
        Cancel = True
    End If
End Sub
```

NOTES:
Disables the close button (1) in the top right corner of the form

```
Private Sub OK_Button_Click()
    Unload fmHelp
End Sub
```

NOTES:
Unloads the *Help* form (Returns the main form *fmAircraft* to view).

DISTRIBUTION LIST

A Replacement Database for the CH-47D Spectrometric Oil Analysis Program

Paul Marsden and Andrew Becker

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor
DGTA (RAAF)
S&T Program
Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
Director General Science Policy Development
} shared copy
Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington (Doc Data Sheet only)
Scientific Adviser to MRDC Thailand (Doc Data Sheet only)
Scientific Adviser Joint
Navy Scientific Adviser
Scientific Adviser - Army
Air Force Scientific Adviser

Aeronautical and Maritime Research Laboratory
Director
Chief of Airframes & Engines Division
Research Leader Propulsion
Albert Wong (AED)
Brian Rebbechi (AED)
Alan Power (AED)
Author(s): Paul Marsden
Andrew Becker

DSTO Library and Archives
Library Fishermans Bend, 1 copy
Library Maribyrnong (Doc Data Sheet only)
Library Edinburgh (Doc Data Sheet only)
Australian Archives
Library, MOD, HMAS Stirling (Doc Data Sheet only)
Library, MOD, Pyrmont (Doc Data sheet only)
US Defense Technical Information Center, 2 copies
UK Defence Research Information Centre, 2 copies
Canada Defence Scientific Information Service, 1 copy
NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

Capability Systems Staff
Director General Maritime Development (Doc Data Sheet only)
Director General Land Development , 1 copy
Director General Aerospace Development, 1 copy

Knowledge Staff

Director General Command, Control, Communications and Computers (DGC4) (Doc Data Sheet only)

Navy

CENGR NAPO, HMAS Albatross, NSW 2540

Army

Commanding Officer, Army Aircraft Logistic Management Squadron, Oakey Aviation Centre, Oakey QLD 4401, (1 copy)

A15 LOGMNGR Army Aircraft Logistic Management Squadron, Oakey Aviation Centre, Oakey QLD 4401, (2 copies)

Stuart Schnaars, ABCA Standardisation Officer, Tobruk Barracks, Puckapunyal, 3662 (4 copies)

SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), MILPO Gallipoli Barracks, Enoggera QLD 4052 (1 copy)

Air Force

CENGR EMU, SRSPO, RAAF Amberley, QLD4306
CENGR MPLMSQN, RAAF Edinburgh, SA 5111
CENGR ALSPO, RAAF Richmond, NSW 2755
CENGR TASPO, RAAF East Sale, VIC 3852

Corporate Support Program

Library Manager, DLS-Canberra
MS Sam Doran
Defence Library Service - Sydney West
Building N157, Defence Establishment
The Northern Road
Orchard Hills NSW 2748

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Hargrave Library, Monash University (Doc Data Sheet only)

OTHER ORGANISATIONS

Mr Alan Yarrow, Innisfail Scientific Services, PO Box 6 Miriwinni, QLD 4871
NASA (Canberra)
AusInfo

OUTSIDE AUSTRALIA

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US
National Aerospace Laboratory, Japan
National Aerospace Laboratory, Netherlands

SPARES (5 copies)

Total number of copies: 55

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE A Replacement Database for the CH-47D Spectrometric Oil Analysis Program	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Paul Marsden and Andrew Becker	5. CORPORATE AUTHOR Aeronautical and Maritime Research Laboratory 506 Lorimer St Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER DSTO-TN-0412	6b. AR NUMBER AR-012-161	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE March 2002

8. FILE NUMBER M1/9/1037	9. TASK NUMBER NAV 00/154	10. TASK SPONSOR DGTA	11. NO. OF PAGES 71	12. NO. OF REFERENCES

13. URL on the World Wide http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0412.pdf	14. RELEASE AUTHORITY Chief, Airframes and Engines Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Oils; spectrochemical analysis; Wear; Military helicopters; Chinook helicopters; Australian Army; Data bases; Computer programs; Failure (mechanics).

19. ABSTRACT

A Spectrometric Oil Analysis Program (SOAP) operates on selected Australian Defence Force platforms to assist in the prediction of incipient machinery failure. Historically, the data from the Australian Army CH-47D helicopters has been stored on a simple Microsoft Excel spreadsheet. DSTO was tasked by the Army Aircraft Logistic Management Squadron to assess the usefulness of this database. This report contains a detailed description of the replacement SOAP database designed by DSTO.

Page classification: UNCLASSIFIED